UNCLASSIFIED

NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
PATUXENT RIVER, MARYLAND

TECHNICAL REPORT

REPORT NO: NAWCADPAX/TR-2004/223

SONOBUOY FIELD DRIFT PREDICTION

by

David S. Hammond

13 January 2005

Approved for public release; distribution is unlimited.

UNCLASSIFIED

DEPARTMENT OF THE NAVY
NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
PATUXENT RIVER, MARYLAND

NAWCADPAX/TR-2004/223
13 January 2005

SONOBUOY FIELD DRIFT PREDICTION

by

David S. Hammond

RELEASED BY:

Michael T. Junod
13 Jan 2005
MICHAEL JUNOD / CODE 4.5.14 / DATE
Head, Acoustic Systems Division
Naval Air Warfare Center Aircraft Division

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE 13 January 2005	2. REPORT TYPE Technical Report	3. DATES COVERED July – December 2004

4. TITLE AND SUBTITLE Sonobuoy Field Drift Prediction	5a. CONTRACT NUMBER N0001404WX20584 N0001405WR10152
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) David Hammond	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Air Warfare Center Aircraft Division 22347 Cedar Point Road, Unit #6 Patuxent River, Maryland 20670-1161	8. PERFORMING ORGANIZATION REPORT NUMBER NAWCADPAX/TR-2004/223
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, Code 32 800 North Quincy Street Arlington VA 22217	10. SPONSOR/MONITOR'S ACRONYM(S) ONR 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A model has been developed that enables the simulation of drift by free-floating buoys within a deployed sonobuoy field. This model, the Sonobuoy Field Drift Model (SFDM), incorporates the results of state of the art primitive equation general circulation models, such as CUPOM and NCOM, in the form of a spatially and temporally varying (4-D) current field. The 4-D current field is used as an input to the model and is the main forcing factor that causes sonobuoy drift. Individual buoy drift response is calculated by extracting vertical current profiles from the current field at the buoy location and specified time, then solving the buoy equilibrium equations in the presence of that current profile. Buoy position is updated after a user defined time interval using the resulting drift vector. This process is applied recursively to the entire buoy field until the end of the simulation time.

Initial comparisons of the model drift predictions to buoy Global Positioning System measured drift data during the LWAD 98-2 exercise have been made. Although buoy-to-buoy comparison of results reveals considerable differences in some instances, in general the SFDM calculated buoy trajectories matched the general behavior of the actual buoy field.

15. SUBJECT TERMS

sonobuoy, sonobuoy field, drift, circulation model, current, hydrodynamic, FF2E, free-floating buoy, Matlab

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON David Hammond
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
Unclassified	Unclassified	Unclassified	SAR	81	301-342-2144

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

SUMMARY

A model has been developed that enables the simulation of drift by free-floating buoys within a deployed sonobuoy field. This model, the Sonobuoy Field Drift Model (SFDM), incorporates the results of state-of-the-art primitive equation general circulation numerical models, such as CUPOM and NCOM, in the form of a spatially and temporally varying (4-D) current field. The 4-D current field is used as an input to SFDM and is the main forcing factor that causes sonobuoy drift. Individual buoy drift response is calculated by extracting vertical current profiles from the current field at the buoy location and specified time, then solving the buoy equilibrium equations in the presence of that current profile. Buoy position is updated after a user defined time interval using the resulting drift vector. This process is applied recursively to the entire buoy field until the end of the simulation time.

Initial comparisons of the model drift predictions to buoy Global Positioning System measured drift data during the LWAD 98-2 exercise have been made. Although buoy-to-buoy comparison of results reveals considerable differences in some instances, the SFDM calculated buoy trajectories matched the general behavior of the actual buoy field.

Contents

List of Figures

ACKNOWLEDGEMENTS

Support for this research was provided by the Office of Naval Research through both the Optimal Deployment of Drifting Acoustic Sensors Departmental Research Initiative managed by Mike Traweek and Manny Fiadeiro, and the Littoral ASW Multistatics Program managed by Larry Green.

The author would like to thank Denny Kirwan, Bruce Lipphardt, and Melissa Zweng from the University of Delaware for assistance with the CUPOM data for the LWAD 98-2 simulations. He would also like to thank the experimental support and programmatic assistance provided by Dave Fenton and the LAMP program, as well as Dick Coughlan, Roger Holler, and Joe McCandless for invaluable technical advice regarding the sonobuoy model. Finally, the author would like to thank Jim McEachern for developing the initial concept for this work and for guiding the development of the model and Walt Farmer, who laid the foundation for this approach over a decade ago.

INTRODUCTION

PROBLEM

The current- and wind-induced drift of sonobuoys can have a detrimental impact on the effectiveness of antisubmarine warfare (ASW) systems utilizing distributed sensor fields. Not only can the sonobuoys drift away from the area-of-interest, but buoy movement relative to other buoys in the field can reduce field integrity by creating coverage gaps in the field or clustering too many buoys in one area. With little relevant experimental data or simulation capability, it is unclear to system planners how severe the drift problem is. A simulation tool needs to be developed to enable the reasonable prediction of sonobuoy trajectories within a deployed field under the influence of realistic temporally and spatially varying current fields.

OBJECTIVE

This effort seeks to develop the capability to simulate the motions of free-floating sensor systems within a distributed sensor field over the period of several days. The successful development of this capability will enable system developers to more effectively evaluate distributed sensor field performance. In addition, the ability to forecast (and hindcast) sonobuoy drift in conjunction with acoustic performance prediction models, will enable operators to select initial deployments such that acoustic coverage will be optimized for the particular mission.

BACKGROUND

It has long been recognized that the drift of sonobuoy systems can be detrimental to the ASW mission. Previous studies (Coughlan, 1976; Holler, 1984; Holler & Scandone, 1987) have shown that field integrity can be compromised due to local spatial and temporal current variations within the field. However, the relatively slow drift and short operating life spans of traditional sonobuoys made drift a manageable problem. In addition, sonobuoys required the presence of a maritime patrol aircraft (MPA) at all times to monitor and record acoustic signals. Sonobuoys that drifted too far from the mission area could be scuttled and replaced by a new sonobuoy from the on-station MPA.

Emerging ASW system concepts seek to develop technologies to make possible the deployment of large distributed fields of off-board sensors (sonobuoys) with operating lives approaching several days. These fields, although deployed from an MPA, would utilize buoys that have an over-the-horizon data link; so constant MPA presence would not be required. For these reasons, the acoustic coverage and overall effectiveness of these systems over multiple day periods is highly dependent on the drift of the individual sensors.

MODEL DESCRIPTION

OVERVIEW

Figure 1 outlines the functional processes involved with the Sonobuoy Field Drift Model (SFDM). The overall model can be categorized as three main procedures: Data Input, Drift Modeling, and Postprocessing.

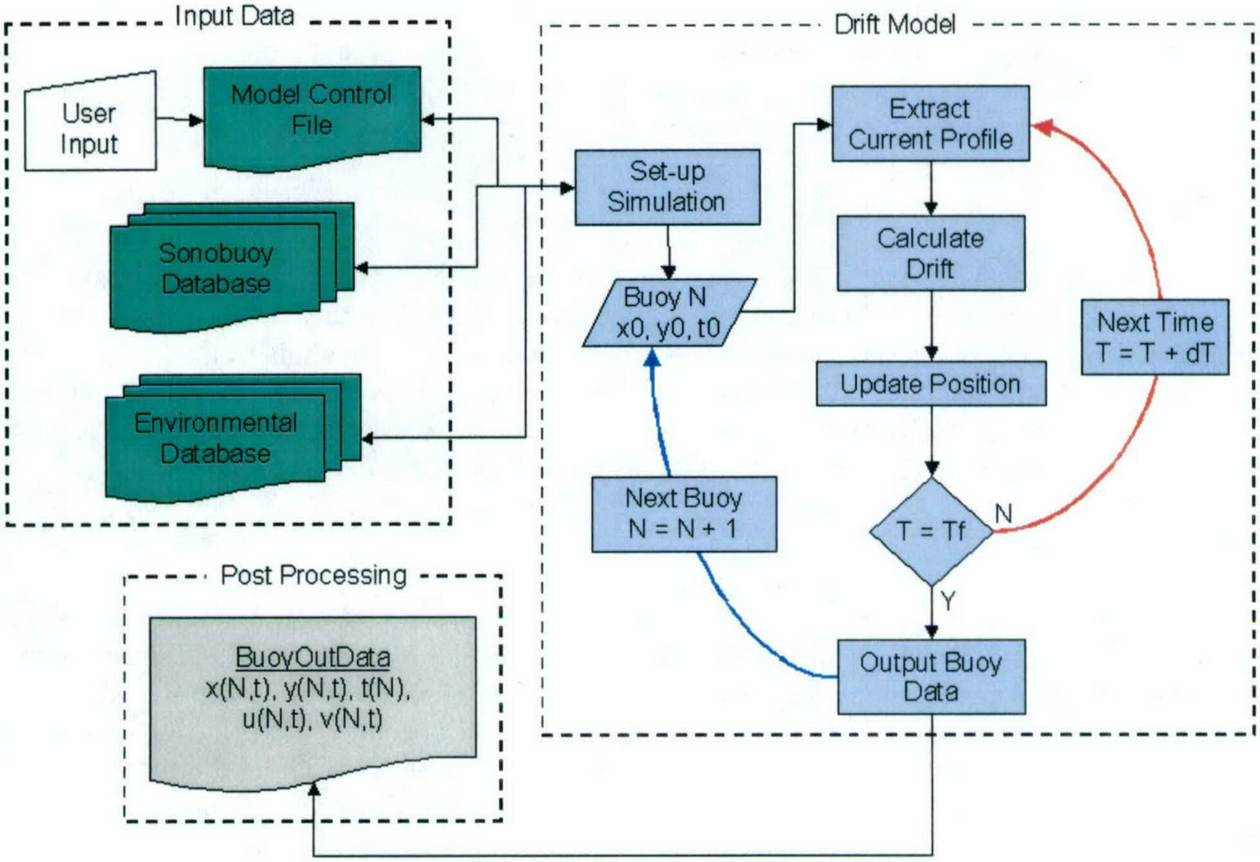


Figure 1: SFDM Functional Diagram

DATA INPUT

An SFDM simulation is built through use of a Model Control File (MCF). The MCF is a Microsoft Excel file with three worksheets: Deployment, Time, and Environment, which define the simulation through user input. An initial sonobuoy deployment pattern is setup in the Deployment worksheet (figure 2). Inputs include sonobuoy type, initial position (latitude and longitude), and date and time of deployment. A plot of buoy positions gives the user a visual check of the position input data. Inputs to the Time worksheet are simulation start date-time, stop date-time, and update time increment. The Environment worksheet contains information about

the current and wind field data including: data file name, grid corner positions and spacing, start and stop date-time, time interval, and depth layers.

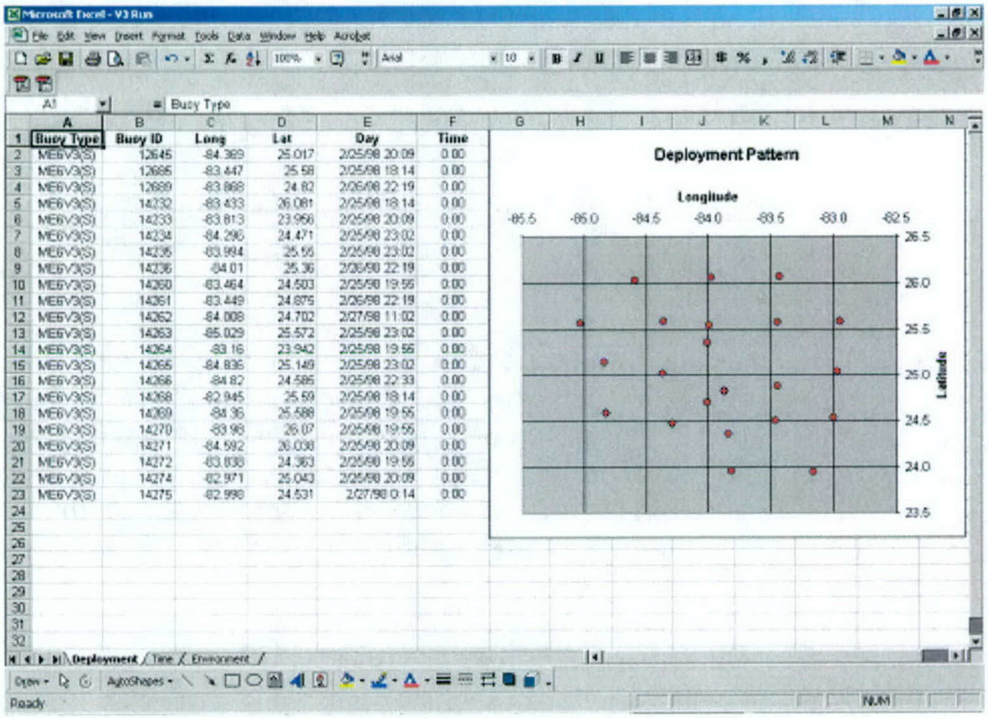


Figure 2: Screenshot of SFDM MCF
(Sonobuoy Deployment Worksheet with Sample Input Data Shown)

The Sonobuoy Database consists of a set of Matlab binary variable files each of which defines the physical characteristics of a particular sonobuoy and depth setting combination. The files contain information pertaining to dimensions, weight, lift and drag of all of the sonobuoy hardware (surface float, cable pack, hydrophone, etc.), cables, and suspension components.

Current and wind information are stored as Matlab binary variable files in the environmental database. Current data files consist of three-dimensional array variables that contain current velocity values in a spatially and temporally varying (4-D) current field. The variable format is as follows:

uN(x position, y position, time) and vN(x position, y position, time),

where,

u and v are the East and North direction current velocities respectively,

N is the depth layer (strata) number defined by the layer depth values in the MCF Environment worksheet,

x and y position are grid coordinates defined by the grid corner longitude and latitude and grid spacing values in the MCF Environment worksheet,

time is the position on the time vector defined by the start/stop time and time interval values in the MCF Environment worksheet.

DRIFT MODELING

SFDM is a collection of Matlab routines that process the input data files, iteratively calculate the drift in time from an initial position of all of the sonobuoys in a field and output the sonobuoy trajectory data for postprocessing.

SFDM first initializes the simulation by processing the MCF data into initialization variables that can be loaded into any of the program modules as needed. The program then enters the first main loop (indicated by the blue arrow in figure 1) by passing the first buoy initial time and position to the current extraction routine. Based on the initial conditions, the current extraction routine develops vertical current profiles using a three-dimensional linear interpolation routine in the u- and v-directions for each current layer. The current profiles are passed to the sonobuoy model to calculate drift velocities in the u- and v- directions.

The sonobuoy model is a modified version of the Navy-standard sonobuoy model FF2E (Wang & Moran, 1971). FF2E is a two-dimensional steady state cable model that is used as a design and evaluation tool by the sonobuoy industry and U.S. Navy sonobuoy developers. It predicts the steady state response (including drift speed) of a free-floating cable-body system (sonobuoy) to a two-dimensional current profile. It has been extensively refined and validated (Houser, 1984; McEachern, 1980; McEachern, 1975) since its initial release. Modifications to FF2E for the SFDM application enable automated processing of FF2E input data (based on sonobuoy type and extracted current profile), and output of FF2E-calculated drift speed in two orthogonal directions (u and v). Appendix A contains the SFDM sonobuoy model (FF2E_D11) program listing that includes details on the specific modifications.

The drift speed values output by the sonobuoy model are combined to obtain a drift vector that is used to calculate the new position of the sonobuoy at the next time step. The time loop (red arrow in figure 1) continues for that sonobuoy: new current profiles are extracted at the new time and position; the sonobuoy model calculates a new drift vector; and the position is updated. This process continues recursively until the simulation stop time is reached, at which time the trajectory data for that sonobuoy is saved in a Matlab cell array variable and the buoy index is incremented so the next buoy can be processed. This continues until trajectories have been developed for all buoys in the field. A complete listing of the SFDM Matlab script is found in appendix B.

POSTPROCESSING

SFDM stores trajectory data in cell array named "buoyOutData" which is saved to a Matlab binary "MAT-file" format with a user specified file name at the end of the routine. The cell array "buoyOutData" contains position vectors, a time vector, and velocity vectors for each buoy in the following format:

buoyOutData { j } = { [x, y, t, u, v] }

where,

j is the sonobuoy index

x and y are the longitude and latitude of the buoy in decimal degrees,

t is the corresponding time in decimal days,

u and v are the North and East velocity magnitude.

The format of the output data is very flexible, allowing customized postprocessing routines to be developed as needed. A Matlab routine called SFDMpost was written to provide some generic plotting and animation routines and provide a template to build more advanced postprocessing capability. SFDMpost can be used to plot single or multiple buoy trajectories against a map background, create an animated sequence of single or multiple buoy motions, and plot single or multiple buoy velocity and heading time histories. A listing of the Matlab script for SFDMpost can be found in appendix C.

THIS PAGE INTENTIONALLY LEFT BLANK

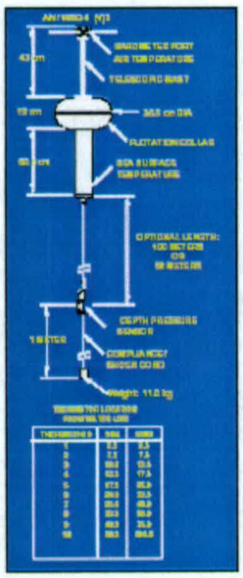
RESULTS

COMPARISON TO LWAD 98-2 DATA

SFDM simulation results were compared to experimental drifting buoy trajectory data from the LWAD 98-2 experiment to access the performance of the model.

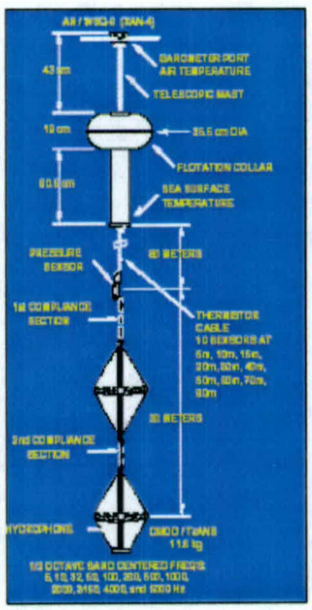
In early 1998, the Naval Research Lab (NRL) conducted a technical evaluation or prototype AN/WSQ-6 drifting oceanographic buoys as part of an LWAD exercise, LWAD 98-2 (Fabre, Koehler, Delgado & Popovich, 1998). Twenty-two AN/WSQ-6(V)3 and seven AN/WSQ-6(V)4 (also referred to as XAN-4) buoys (figure 3) were deployed in the Gulf of Mexico on 23 February 1998 by Navy marine patrol aircraft. The WSQ-6 series buoys are capable of measuring and transmitting, via ARGOS satellite link, a variety of oceanographic data and Global Positioning System (GPS) location for 30 to 60 days. The WSQ-6 GPS data from this exercise was used to compare to SFDM calculated trajectories.

AN/WSQ-6(V)3



- Measured Parameters: BP, AT, SST, Tz (to 50m or 100m), Drift
- Air descent mechanism (includes windflap and parachute)
- Upper unit and float assembly (includes salt water detection for activation, data processing, electronics, transmitter, sensors and power supply)
- Thermistor cable (Tz)(includes subsurface temperature sensors and cable) which measures and records temperatures at 2.5, 7.5, 10.0, 12.5, 17.5, 20.0, 25.0, 32.5, 40.0, 50.0 meters
- ARGOS transmitter

AN/WSQ-6 (XAN-4)



- Measured Parameters: BP, AT, SST, AN (5 Hz to 5 kHz), Tz (80m), Drift
- Air descent mechanism (includes windflap and parachute)
- Upper unit and float assembly (includes salt water detection for activation, data processing, electronics, transmitter, sensors and power supply)
- Thermistor cable (Tz)(includes subsurface temperature sensors and cable) which measures and records temperatures at 2.5, 7.5, 10.0, 12.5, 17.5, 20.0, 25.0, 32.5, 40.0, 50.0 meters
- ARGOS transmitter

Figure 3: AN/WSQ-6 Specifications

This initial assessment focused on the WSQ-6(V)3 model, because the more complicated drag characteristics of the WSQ-6 XAN-4 buoy were unknown. Inputs for a (V)3 sonobuoy model were developed based on the dimensions shown in figure 3 (the only data available at the time).

As part of previous investigations into the circulation dynamics of the Gulf of Mexico (Kirwan, Toner, & Kantha, 2003; Toner, Kirwn, Poje, Kantha, Muller-Karger & Jones, 2003), current field data modeled using the University of Colorado version of the Princeton Ocean Model (CUPOM) covering the LWAD 98-2 test site and dates was available. The data were provided by the University of Delaware and formatted for input to SFDM. Horizontal resolution was 1/12 deg, vertical layers were at 0, 10, 20, 30, 50, 75 and 100 m and the time interval was 24 hr.

After the WSQ-(V)3 model was added to the sonobuoy database and the CUPOM LWAD 98-2 data were set up in the environmental database, a SFDM simulation of the (V)3 trajectories was constructed. Initial positions were determined from the initially reported buoy GPS data. Time parameters were set for a 4-day simulation run time with 1-hr position updates. Figure 4 plots the GPS data from the LWAD 98-2 exercise (blue) and the SFDM results (red). Based on the GPS data, the field can be divided into three regions based on the general drift characteristics of the buoys: a northwest region, southwest region, and northeast region. The gray dashed lines in figure 4 indicated the boundaries of the three regions and the blue dotted line indicates the 50-fathom (328 m) contour line. The partition into three regions is in general agreement with Toner (2002). According to Toner, particular material curves, or inflowing and outflowing manifolds, delineate these regions, and each region will be characterized by distinct buoy drift patterns.

Although individual trajectories predicted by the SFDM did not exactly overlay the corresponding GPS data, the general regional nature of the field drift was forecast correctly. Buoys in the northwest region [12, 19] transited in a southwest direction. The Loop Current and Gulf Stream dominated the buoys in the southwest region [1, 3, 5, 6, 9, 11, 13, 15, 20], causing the buoys to drift in a generally southeast direction at a relatively high speed. Buoys deployed in the northeast region [2, 4, 7, 16, 18, 21, 22] loitered near the area they were initially deployed. Three buoys deployed near regional boundaries [8, 10, 17] were observed to have anomalous drift behavior. Buoys 8 and 17 drifted slowly in a northerly direction and GPS data from buoy 10 indicated a slow southerly drift. In all three cases, the SFDM results differed significantly from the GPS data.

Table 1 lists the average drift speed and heading data from the WSQ-6(V)3 GPS data (measured) and SFDM (modeled) and the corresponding percent error. The data are color-coded according to region.

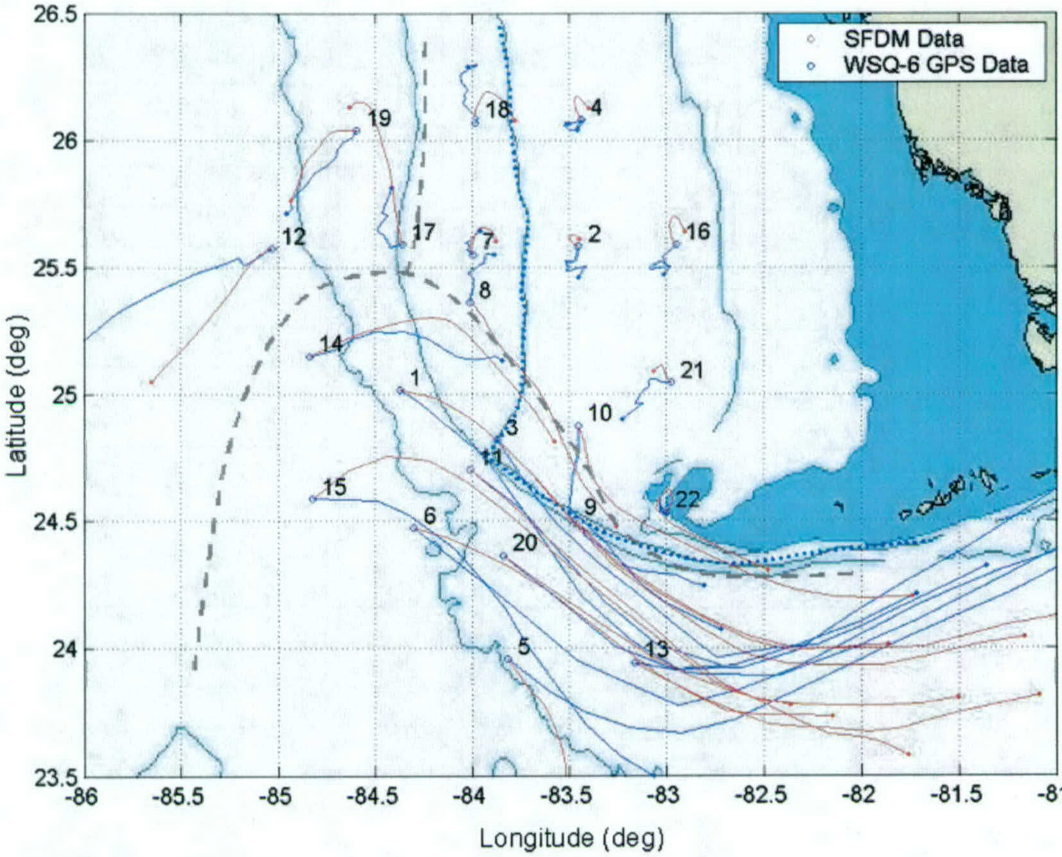


Figure 4: Comparison of WSQ-6(V)6 GPS Data from the LWAD 98-2 Exercise (blue) and SFDM Results (red)

Table 1: Comparison of LWAD 98-2 WSQ-6(V)3 GPS Data (measured) with SFDM Simulation Results (modeled)

Buoy ID	Drift Speed (m/s)			Heading (deg)		
	Measured	Modeled	Error	Measured	Modeled	Error
12	0.74	0.15	-80%	-69.6	-138.2	99%
19	0.17	0.19	12%	-106.1	-127.7	20%
1	1.03	0.92	-11%	110.1	118.1	7%
3	0.96	1.03	7%	101.4	125.1	23%
5	1.17	0.26	-78%	83	88.1	6%
6	1.2	0.7	-42%	95.2	118	24%
9	0.25	0.95	280%	113.8	103	-9%
11	1.22	1.04	-15%	121.5	124.9	3%
13	1.55	0.62	-60%	56	102	82%
14	0.35	0.49	40%	91.7	100.9	10%
15	0.72	0.92	28%	120.6	108.6	-10%
20	1.35	0.75	-44%	80.2	110.6	38%
2	0.13	0.04	-69%	-31.4	93.2	-397%
4	0.13	0.06	-54%	-29.6	61.7	-308%
7	0.06	0.07	17%	-21.5	75.6	-452%
16	0.13	0.06	-54%	-63.5	57	-190%
18	0.14	0.11	-21%	-16.6	91.3	-650%
21	0.13	0.04	-69%	-72.1	-48.3	-33%
22	0.1	0.11	10%	21.2	92.8	338%
8	0.11	0.21	91%	10.3	134.5	1206%
10	0.42	0.52	24%	-25.4	139.8	-650%
17	0.13	0.25	92%	-13.5	-19.3	43%

Northwest
Southwest
Northeast
Border

Typical and anomalous individual buoy trajectories from each region are presented in appendix D. For brevity, plots of buoys with noticeably similar results will be omitted.

DISCUSSION

LOOP CURRENT LOCATION

With the exception of a few anomalous results, the SFDM predictions represent the general drift characteristics of the LWAD 98-2 buoy field; however, it appears that the CUPOM data predicted the Loop Current to lie further North that indicated by the WSQ-6 GPS data. This is evidenced by comparing the behavior and predicted behavior of buoys 5, 8, 9, and 10 (refer to figures D-7 through D-12 and D-16 through D-21). The actual drift of buoy 5 (southernmost of these four buoys) is indicative of a buoy under the influence of the Loop Current/Gulf Stream: high average drift speed with a heading that is southeast turning east-northeast at a location east of –83 deg longitude (example buoy 3, figures D-4, D-5, and D-6). However, the predicted path of buoy 5 takes it towards the coast of Cuba – obviously not effected by the Loop Current. Conversely, GPS data from the more northerly buoys 8, 9, and 10 indicate little effect from the Loop Current (at least initially), but the SFDM data shows a strong Loop Current effect. This is likely the reason that the SFDM drift velocity results for the southerly buoys in the southwest region are typically less than the GPS data while the northerly buoys have a higher predicted drift speed.

If the buoys near the edges of the Loop Current are discarded, the average drift speed error for the southwest region is 29% and the average heading error is 25%.

SHALLOW WATER LIMITATIONS

According to Toner (2002), there are limitations in the CUPOM model in shallow water (less than 100 m). In addition, the CUPOM modeled data does not include tidal effects. This helps explain some of the differences observed between the SFDM results and the GPS data in the northeast region (typical northeast results shown in figures D-13, D-14, and D-15). Although the model correctly predicted the generally low drift speed of the buoys, the predicted headings are significantly different. All of the buoys in the northeast region have periodic fluctuations in heading that appear to have a period roughly coinciding with tidal periods. As this tidal effect was not part of the modeled current field and the circulation model becomes less reliable in shallow water, it was impossible to replicate the northeast region trajectories accurately.

NORTHWEST REGION DRIFT

The overall drift of buoys and the CUPOM current field in the northwest region was correctly predicted to be in a predominantly southwesterly direction. This demonstrates the strength of data assimilating general circulation models such as CUPOM used by the SFDM. It is a prediction that probably would not have been evident to operational personnel prior to deployment. The ability to simulate unanticipated buoy field behavior, like that observed in LWAD 98-2, is an asset to mission planners.

2-D DRIFT ERROR

Real world current fields typically have a complex three dimensional structure, which, in turn, produces a three dimensional response from a drifting buoy supporting a relatively complex configuration of cable, subsurface components, and acoustic sensors (sonobuoy). The SFDM model currently uses a modified version of the sonobuoy model FF2E to calculate the sonobuoy response. As discussed in a previous section, FF2E is restricted to two dimensions, so only planar current profiles can be used to calculate buoy response. FF2E was considered a good model to use for the sonobuoy drift response module of SFDM because; one, it is an accepted Navy simulation tool; two, the CUPOM data format – current is described by North and East current velocities – lends itself to planar current profile extraction; and three, this enabled the quickest development path for SFDM as minimal modification to the original source code was required.

Calculating sonobuoy response to a three-dimensional current profile by representing the current as two orthogonal planar current profiles does, however, cause some error. Drag on an object can be characterized by the equation:

$$D = \tfrac{1}{2}\ \rho\ Cd\ A\ \mathbf{V}^2,$$

where,

D is the drag,

ρ is the fluid density,

Cd is the object coefficient of drag,

A is the object cross-sectional area,

V is the magnitude of the fluid relative velocity past the object.

V can be described by two orthogonal components, u and v. SFDM calculates drag on various components first in the u direction then in the v direction, such that the drag error is (full derivation can be found in appendix E):

$$E_D = \sqrt{\sin^4\theta + \cos^4\theta}$$

where,

E_D is the ratio of drag calculated by the u and v components over the drag calculated using **V**,

θ is the direction of the fluid relative current: $\theta = \text{atan}\ (u / v)$.

This error, plotted in figure 5, indicates that drag can be under predicted by almost 30% at an angle of 45 deg.

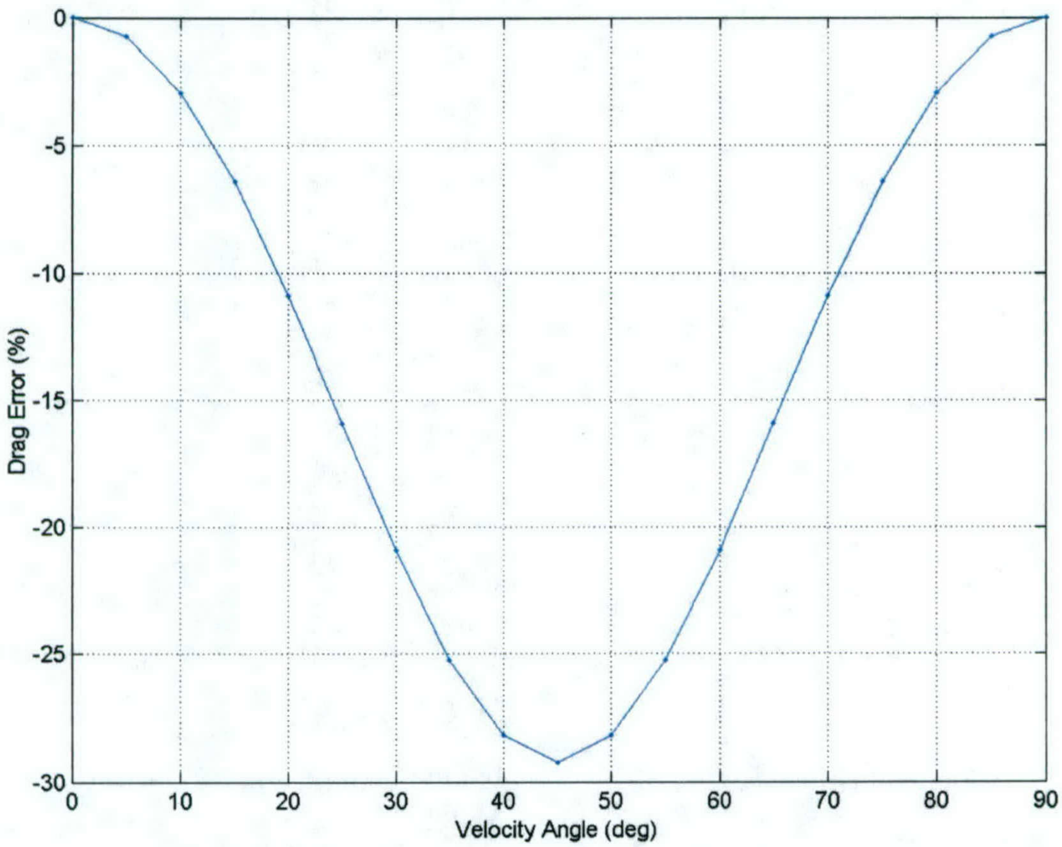


Figure 5: Drag Error Plotted as Velocity Direction Changes from North (0 deg) to East (90 deg)

Drag, and ultimately drift response, of a sonobuoy is governed by the same principals as the simple drag error calculated above. In order to reduce this error, SFDM was modified (version 2.5) such that the two orthogonal axes describing the current were rotated to align with the predominant current direction, determined by a weighted mean (see appendix E for a more detailed description). Thus, the new current components, u' and v', represented a prevailing current profile and a lesser profile. The resulting drift vectors were then transformed back into the global North and East coordinate system. The effectiveness of this approach was tested by rotating a simple planar current profile between 0 deg to 90 deg, and comparing the calculated drift at each current direction to the drift at 0 (or 90) deg (the assumed correct drift). This drift error is plotted in figure 6 for the North-East coordinate system approach (SFDM version 2.0) and the prevailing-minor coordinate system approach (SFDM version 2.5). For this simple case, the prevailing-minor approach reduces drift error to zero.

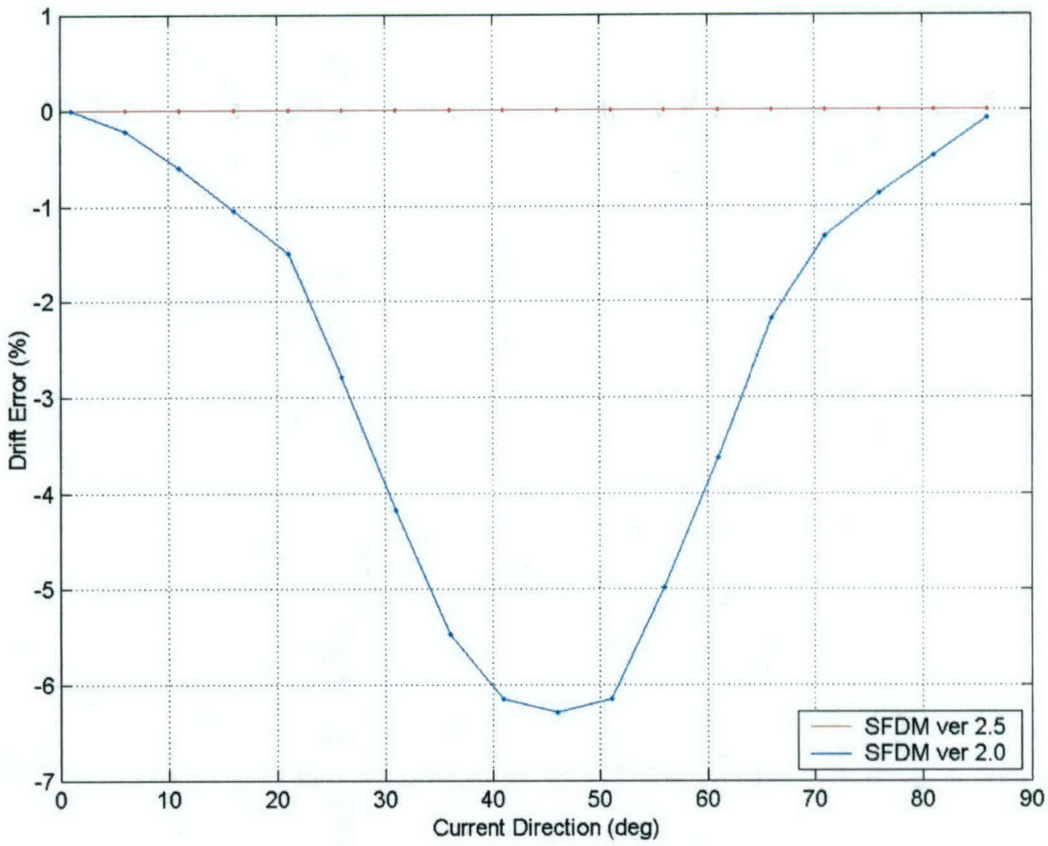


Figure 6: Drift Error Comparison between SFDM Version 2.0 and 2.5

Further investigations were conducted to determine the drift error relationship to current profile vertical shear and buoy drag distribution; and a comparison between SFDM version 2.0 and 2.5 simulations using the LWAD 98-2 data were performed (Hammond, 2004). There were no substantial differences in the LWAD 98-2 simulations probably due to the low subsurface drag of the WSQ-6 buoys and the low shear current profiles in the area.

RECOMMENDATIONS

SONOBUOY MODEL IMPROVEMENTS

It is apparent from the Drift Error study results that small errors can arise when a two-dimensional sonobuoy model, such as FF2E, is used to calculate sonobuoy drift response in a three-dimensional current field. These errors accumulate over time and degrade the effectiveness of the overall sonobuoy field drift model. A three-dimensional steady state sonobuoy model should be developed and incorporated into SFDM to account for the complexity of real world current fields.

Current fields also have a temporal component. A dynamic sonobuoy model that calculates the time varying response of the sonobuoy can be developed and incorporated into SFDM; however, given the typical spatial and temporal resolution of the current velocity field data, the development of a three-dimensional dynamic sonobuoy model will probably not improve the overall performance of SFDM significantly.

ENVIRONMENTAL DATABASE

The CUPOM data used to simulate the LWAD 98-2 experiment had a spatial resolution of 1/12 deg and a temporal resolution of 24 hr. Increasing the spatial and temporal resolution of the current field data would enhance capability of SFDM. Similarly, the inclusion of tidal data in the current field would enhance SFDM.

The model is currently capable of incorporating wind field data; however, this function has not been tested. Investigations should be made into the availability and format of wind data.

VALIDATION TESTING

Models must be validated before they can be confidently used for simulations. While SFDM validation can be accomplished by simply tracking the position of free-floating sonobuoys, true validation would also require measurement of the environment (current, wind, and waves) at the sonobuoy. An effort should be made to conduct a full validation of the SFDM, both in physically well-understood areas and operationally significant areas.

In October 2004, the LAMP program deployed 12 Davis drifter buoys in a location of operational significance. This data will be used to continue validation of the general circulation model used by the Navy in this area. LAMP and the ODDAS programs are committed to supporting validation efforts for sonobuoy drift modeling.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

1. Faber, J.P., Koehler, K.A., Delgado, R., Popovich, W. (1998, Jul). Demonstration and evaluation of AN/WSQ-6 drifting meteorology and oceanographic buoys during the Littoral Warfare Advanced Development (LWAD) 98-2 test (NRL 7140-98-002). Washington, DC: Naval Research Lab.

2. Hammond, D.S. (2004, Nov). SFDM drift error study (NAWCAD Technical Note 4514-003-2004). Patuxent River, MD: Naval Air Warfare Center Aircraft Division.

3. Holler, R.A. (1984, Apr). Sonobuoy drift control experiments near Bermuda (NADC 3043-TM-111-84). Warminster, PA: Naval Air Development Center.

4. Holler, R.A., Scandone, C. (1987, Jan). Drift and dispersion of sonobuoy fields near Hawaii (NADC 5043-TM-160-87). Warminster, PA: Naval Air Development Center.

5. Houser, K.L. (1984, May). Update to the free floating extensible cable system model (FF2E) (NAC-TR-2359). Indianapolis: Naval Avionics Center. (DTIC No. ADA144369).

6. Kirwan, A.D., Toner, M., Kantha, L. (2003). Predictability, uncertainty, and hyperbolicity in the ocean. *International Journal of Engineering Science*, *41*, 249-258.

7. McEachern, J.F. (1975, November). SVLA array cable drage evaluation (NADC 2063-TM-30-75). Warminster, PA: Naval Air Development Center.

8. McEachern, J.F. (1980, May). A modification to the free floating extensible cable system computer model (FF2E) to consider lift and drag forces on intermediate bodies (NADC-80178-30). Warminster, PA: Naval Air Development Center. (DTIC No. ADA092512)

9. Toner, M. (2002). Post-analysis of: AN/WSQ-6 Navy drifting buoy program testing in the Littoral Warfare Advanced Development (LWAD) 98-2 sea test.

10. Toner, M., Kirwan, A.C., Poje, A.C., Kantha, L.H., Muller-Karger, F.E., Jones, C.K.R.T (2003). Chlorophyll dispersal by eddy-eddy interactions in the Gulf of Mexico. *Journal of Geophysical Research, 108(C4),* 3105-3127.

11. Wang, H.T., Moran, T.L. (1971, Oct). Analysis of the two-dimensional steady-state behavior of extensible free-floating cable systems (NSRDC Report No. 3721). Bethesda, MD: Naval Ship Research and Development Center.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
SONOBUOY MODEL SOURCE CODE LISTING (FF2E_D11)

```
C ==================================================================
      PROGRAM FF2E_D10
C ==================================================================
C
C               FF2E DRIFT VER 1.0
C
C  DESCRIPTION:
C   This is a modified version of FF2E written specifically for
C   the sonobuoy field drift model (SFDM). Input files are
C   created by SFDM and an output file passes drift data to SFDM
C
C  MODIFICATIONS:
C
C   VER 1.0 / DAVE HAMMOND / NAVAIR 4.5.14.2 / 29 OCT 2004
C
C   1. INPUT DATA FILE = "IN.DAT" (CREATED BY SFDM) - USES
C      SAME FORMAT AT PREVIOUS VERSIONS OF FF2E
C   2. NCASES = 2
C   3. Output file created: "DRIFT.DAT"
C      Contains x and y direction drift data
C   4. Enabled negative drift handling with IFLIP parameter
C      If the first current profile velocity is negative,
C      reverse the direction of the current profile and run.
C      IFLIP reverses the drift direction before writing to
C      "DRIFT.DAT".
C   5. Added "DEBUG.DAT" File (replace output file)
C   6. Fixed error that occurs if last current profile depth
C      layer is less than the length of the buoy by setting
C      all velocities below that point = YYK(NCUR)
C
C   VER 1.1 / DAVE HAMMOND / NAVAIR 4.5.14.2 / 6 NOV 2004
C
C   1. Fix problem with negative CREL velocities and VAR sub
C      by reversing CREL if it is less than zero before calling
C      VAR subroutine, then changing it back afterwards.
C
C
C
C  REFERENCES:
C   WANG, H.T., "ANALYSIS OF THE TWO-DIMENSIONAL STEADY-
C     STATE BEHAVIOR OF EXENSIBLE FREE-FLOATING CABLE SYSTEMS",
C     NSRDC REPORT 3721, OCT 1971
C
C   MCEACHERN, J.F., "A MODIFICATION TO THE FREE FLOATING
C     EXTENSIBLE CABLE SYSTEM (FF2E) TO CONDSIDER LIFT AND DRAG
C     FORCES ON INTERMEDIATE BODIES", NADC REPORT 80178-30,
C     MAY 1980
C
C   HOUSER, K.L., "UPDATE TO THE FREE FLOATING TWO-DIMENSIONAL
C            EXTENSIBLE CABLE SYSTEM MODEL (FF2E)", NAC REPORT TR-2359,
C     MAY 1984
C
C -------------------------------------------------------------------
C  Variable Declaration
C -------------------------------------------------------------------
```

```
      IMPLICIT REAL (A-H,O-Z)
      REAL L,LA,LB
      DIMENSION CVFAC(100),CDINIT(10),DDRAG(75),DLIFT(75)
C
      COMMON /BLK1/ DB,DA,LB,LA,WB,CDB1,CDB2,FTANG,UDRIFT,H,DELTAS,
     1PRINTI,UWIND,CDA,EPSLON,TBH,TBV,NCAB,NHPHS,CDAPK,WPAK,EP2
      COMMON/BLK2/XX(30),YY(30),NPROF,FIRST,RHO,NCUR
      COMMON/BLK4/DRAG,WPLA,WPLB,FFTANG,DRIFT,TREFC,AEC,PC
      COMMON/BLK5/NPR(100),DC(100),WC(100),FLC(100),CDC(100),
     1TREF(100),P(100),CDABD(100),WBD(100),DCI(100),WCA(100),
     2WCB(100),YYK(30)
      COMMON/BLK6/PHIM(10,7),U(10,10),L(10,10,7),D(10,10,7),NBOD(100),
     1NPHI(10),NU(10)
      COMMON/BLK7/FAE(100,15),AE(100,15),NAE(100),JAM
      COMMON/BLK8/NFOSB,FOSB(16),VOSB(16)
      COMMON/BLK9/NCTR
C -----------------------------------------------------------------
C  Format Statements
C -----------------------------------------------------------------
2     FORMAT(8F10.6)
3     FORMAT(8F10.4)
4     FORMAT(F12.4,4F12.6,I3)
5     FORMAT(2F12.6,F12.4,F12.6,I3)
6     FORMAT(F10.3)
8     FORMAT(20I4)
44    FORMAT(8F10.2)
110   FORMAT(A80)
222   FORMAT(A20)
C -----------------------------------------------------------------
C  Open input and output files
C -----------------------------------------------------------------
      OPEN(5,FILE="IN.DAT")
      OPEN(4,FILE="DRIFT.DAT")
C      OPEN(6,FILE="DEBUG.DAT")
C -----------------------------------------------------------------
      DO 870 ICASE=1,2
C -----------------------------------------------------------------
C  Read data from input file
C -----------------------------------------------------------------
      READ(5,8)NCUR,NCAB,NTAB,NFOSB,NHPHS,NCTR
      NPROF=NCUR
      READ(5,2) DAI,LA,CDA,TBH,TBV
      READ(5,2) DB,LB,CDB1,CDB2,WB,UWINDK
      IF(NFOSB.EQ.0) GOTO 10
      READ(5,3)  (FOSB(K),K=1,NFOSB)
      READ(5,3)  (VOSB(K),K=1,NFOSB)
10    READ(5,2)CDAPK,WPAK,RHO
      IF(RHO.EQ.0.0) RHO=1.9905
      READ(5,4)(FLC(K),DCI(K),WC(K),CDC(K),CVFAC(K),NPR(K),
     1K=1,NCAB)
      READ(5,8)(NAE(K),K=1,NCAB)
      DO 100 NN=1,NCAB
      NE=NAE(NN)
      READ(5,3)(AE(NN,LL),LL=1,NE)
      IF(NE.EQ.1) GOTO 100
      READ(5,3)(FAE(NN,LL),LL=1,NE)
100   CONTINUE
      READ(5,5)(WBD(K),CDABD(K),TREF(K),P(K),NBOD(K),
     1 K=1,NCAB)
      IF(NCUR.LE.1) GOTO 101
```

```
      READ(5,44) (XX(K),K=1,NCUR)
      READ(5,2) (YYK(K),K=1,NCUR)
101   IF(NTAB.LE.0) GOTO 380
      DO 16 N=1,NTAB
      SGU = 0.
      READ(5,8) NPHI(N),NU(N)
      NPT=NPHI(N)
      NUT=NU(N)
      READ(5,3) (PHIM(N,I),I=1,NPT)
      READ(5,3) (U(N,J),J=1,NUT)
      NP=NPHI(N)*NU(N)
      READ(5,2) (DDRAG(M),M=1,NP)
      READ(5,2) (DLIFT(M),M=1,NP)
C ------------------------------------------------------------------
C  Set up lift and drag variables from tabulated data
C ------------------------------------------------------------------
      DO 12 I=1,NUT
      SGU=SGU+U(N,I)
      DO 12 J=1,NPT
      INDX=J+(I-1)*NPHI(N)
      D(N,I,J)=DDRAG(INDX)
12    L(N,I,J)=DLIFT(INDX)
16    CONTINUE
C ------------------------------------------------------------------
C       Compute average body CdA from tabulated body data
C ------------------------------------------------------------------
      SGD=0.
      SGUF= SGU*1.688/NU(N)
      DO 370 ID=1,NP
370   SGD =SGD+DDRAG(ID)
      SGDA=SGD/NP
375   CDINIT(N)=2.*SGDA/(1.9905*SGUF**2)
380   EPSLON=0.0001
      EP2=0.0001
      FTANG=.020
      DA=DAI/12.0
      DO 395 J=1,NCAB
      DC(J)=DCI(J)/12.
C ------------------------------------------------------------------
C  Assign initial CdABD from avg. tabulated drag and velocity data
C ------------------------------------------------------------------
      IF(NBOD(J).LE.0)GO TO 395
      CDABD(J)=CDINIT(NBOD(J))
395   CONTINUE
C ------------------------------------------------------------------
C  If the first current profile velocity is negative, reverse
C  the direction of the current profile and set IFLIP = 1
C ------------------------------------------------------------------
      IFLIP=0
      IF(YYK(1).GT.0.) GOTO 397
      IFLIP=1
      DO 396 I=1,NCUR
396   YYK(I)=-YYK(I)
C ------------------------------------------------------------------
C  Find the maximum and minimum values of the current
C ------------------------------------------------------------------
397   TOTL=0.
      DO 400 J=1,NCAB
400   TOTL=TOTL+FLC(J)
      TOTL=1.3*TOTL
```

```
      UMAXK=-1000.
      UMINK=1000.
      DO 405 I=1,NCUR
      IF(YYK(I).GT.UMAXK) UMAXK=YYK(I)
      IF(YYK(I).LT.UMINK) UMINK=YYK(I)
      IF(XX(I).GT.TOTL) GO TO 410
405   CONTINUE
C ---------------------------------------------------------------------
C  Convert from knots to ft/s
C ---------------------------------------------------------------------
410   UMAX=1.688*UMAXK*1.2
      UMIN=1.688*UMINK*.8
      UWIND=1.688*UWINDK
      DO 845 I=1,NCUR
      YY(I)=1.688*YYK(I)
845   CONTINUE
      IF(NTAB.LE.0)GO TO 860
      DO 855 N=1,NTAB
      NT=NU(N)
      DO 855 K=1,NT
      U(N,K)=1.688*U(N,K)
855   CONTINUE
C ---------------------------------------------------------------------
C  Make CVFAC cable adjustments
C ---------------------------------------------------------------------
860   DO 865 J=1,NCAB
      ALPHA=CVFAC(J)*0.25*64.043*3.14159
      WCB(J)=ALPHA*DC(J)*DC(J)
      WCA(J)=WC(J)+WCB(J)
865   CONTINUE
      IRUN=1
C ---------------------------------------------------------------------
C  Call the subroutine STEADY to calculate the cable configuration
C ---------------------------------------------------------------------
      CALL STEADY(IRUN,UMAX,UMIN,IFLIP)
C ---------------------------------------------------------------------
C                 End of main routine loop (lable 870)
C ---------------------------------------------------------------------
870   CONTINUE
C ---------------------------------------------------------------------
C  Close files and end program
C ---------------------------------------------------------------------
      CLOSE(5)
      CLOSE(6)
      STOP
      END
C =====================================================================
C                         End of main program
C =====================================================================
C
C
C =====================================================================
C                          ** SUBROUTINES **
C =====================================================================
C
C     Subroutine STEADY
C
C   This routine calculates the steady state configuration
C
C =====================================================================
```

```
      SUBROUTINE STEADY(IRUN,DLIMIT,UDMIN,IFLIP)
C -----------------------------------------------------------------
C  Variable Declaration
C -----------------------------------------------------------------
      DIMENSION CRELKT(400)
      DIMENSION FLIFT(10),BDRAG(10)
      DIMENSION S(400),X(400),Y(400),PHI(400),PHID(400),T(400)
      DIMENSION XX(400),YY(400),Y0(5),PHIV(400),BPHI(400),SE(400)
      DIMENSION BPHIV(400),SAR(400)
      COMMON /BLK1/ DB,DA,LB,LA,WB,CDB1,CDB2,FTANG,UDRIFT,H,DELTAS,
     1PRINTI,UWIND,CDA,EPSLON,TBH,TBV,NCAB,NHPHS,CDAPK,WPAK,EP2          COMMON
/BLK3/ FIRST
      COMMON /BLK4/ DRAG,WPLA,WPLB,FFTANG,DRIFT,TREFC,PC
      COMMON /BLK5/ NPR(100),DC(100),WC(100),FLC(100),CDC(100),
     1TREF(100),P(100),CDABD(100),WBD(100),DCI(100),WCA(100),
     2WCB(100),YYK(90)
      COMMON /BLK6/ PHIM(10,7),U(10,10),L(10,10,7),D(10,10,7),NBOD(100),
     1NPHI(10),NU(10)
      COMMON /BLK7/ FAE(100,15),AE(100,15),NAE(100),JAM
      COMMON /BLK8/ NFOSB,FOSB(15),VOSB(15)
      REAL LA,LB
C -----------------------------------------------------------------
c  Constants
C -----------------------------------------------------------------
      DATA PI,RHO,RHOAIR,GAMMA,RADIAN
     1/3.14159,1.9905,.002378,64.043,57.29578  /
C -----------------------------------------------------------------
C  Solution status format statements
C -----------------------------------------------------------------
970   FORMAT(1X,42HREVERSAL IN SIGN BETWEEN DELTAU AND ERRORH)
975   FORMAT(1X,28HSTART OF SIMULTANEOUS SCHEME)
980   FORMAT(1X,25HSTART OF STAGGERED SCHEME)
C -----------------------------------------------------------------
C  Set iteration limits / initial parameters
C -----------------------------------------------------------------
      DLLIMIT=DLIMIT
      UDDMIN=UDMIN
      WBOT=WBD(NCAB)
      GPRBSQ=GAMMA*PI*.25*DB*DB
      GPIOV4=GAMMA*PI*.25
      ILAST=0
      HMIN=WB/GPRBSQ
      UDRIFT=UDDMIN+0.5*(DLIMIT-UDDMIN)
      DLIMIT=1.01*DLLIMIT
C -----------------------------------------------------------------
C  Let initial buoyancy be the weight of everything under the buoy
C -----------------------------------------------------------------
      XNPHS=NHPHS
      DELTA=1.
      BCY=0.
      DO 1000 J=1,NCAB
      BCY=BCY+FLC(J)*WC(J)+WBD(J)
1000  CONTINUE
      BCY=BCY+WPAK+TBV
      IF(BCY.LE.0.) BCY=0.
      H=(BCY+WB)/GPRBSQ
      UMAX=DLIMIT
      UMIN=UDDMIN
      UMIN1=UMIN
      UMAX1=UMAX
```

```
      PRV=0.
      PRH=0.
      ABSERH=0.
      ABSERV=0.0
      PERH=15.
      PERV=15.
      EPRIME=100.
      BRSLT=EPRIME
      DEN4=LB*CDB1*DB+CDAPK
      DO 1005 J=1,NCAB
      DEN4=DEN4+CDC(J)*DC(J)*FLC(J)
      DEN4=DEN4+CDABD(J)
1005  CONTINUE
      DEN5=RHO*DEN4
      IB=0
      USEN=1.
      K2=0
      K3=0
      I2MANY=0
      IRUN=1
      INO=1
      KIT=0
      KUSTOP=100
      IFRST=11
      KUD=0
      KH=0
      KREV=0
      DHFAC=0.8
      PRERV=1000.
      HMINP=HMIN
      HMAXP=LB
      F=0.0
1010  CONTINUE
      IF(H.LT.HMIN) H=1.01*HMIN
      BCY=GPRBSQ*H-WB-WPAK
      HTEMP=H
      FIRST=-100.0
      UDKNTS=.5924*UDRIFT
      FIRST=100.0
C ---------------------------------------------------------------
C  Calculate wind drag
C ---------------------------------------------------------------
      DRAGW=.5*RHOAIR*DB*CDB2*(UWIND-UDRIFT)*ABS(UWIND-UDRIFT)*(LB-H)
     1+.5*RHOAIR*CDA*(UWIND-UDRIFT)*ABS(UWIND-UDRIFT)*LA*DA
C ---------------------------------------------------------------
C  Calculate drag on surface float package
C ---------------------------------------------------------------
      CALL CUR(H,COFY)
      CRELP=COFY-UDRIFT
      DRGPK=0.5*RHO*CDAPK*CRELP*ABS(CRELP)
C ---------------------------------------------------------------
C  Find surface float drag
C   (if surface float drag table exists iterate Cd until drag
C   error very small)
C ---------------------------------------------------------------
      DEP=0.5*H
      CALL CUR(DEP,COFY)
      CZERO=COFY
      CREL=COFY-UDRIFT
1015  DRAGB=.5*RHO*DB*CDB1*CREL*ABS(CREL)*H
```

```
      DRAGB=DRAGB+DRGPK
      IF(NFOSB.EQ.0) GOTO 1020
C
      IF(CREL.LT.0.) CREL = -CREL
C
      CALL VAR(DRAGB,CREL,H,DB,CDB1,CD)
C
      IF(CREL.LT.0.) CREL = -CREL
C
      IF(CD.EQ.CDB1) GOTO 1020
      CDB1=CD
      GOTO 1015
1020  DRAGB=DRAGB+DRAGW
C ----------------------------------------------------------------------
C  INITIAL TENSION IS THE RESULTANT OF BUOYANCY AND DRAG.
C ----------------------------------------------------------------------
      T(1)=SQRT(BCY*BCY+DRAGB*DRAGB)
C ----------------------------------------------------------------------
C  INITIAL ANGLE IS THE ANGLE WHOSE TANGENT IS BUOYANCY/DRAG OF BUOY.
C ----------------------------------------------------------------------
      PHI(1)=ATAN2(BCY,DRAGB)
      X(1)=0.0
      PHID(1)=PHI(1)*RADIAN
      S(1)=0.0
      SE(1)=0.0
      Y(1)=H
      NLAST=0
      DRIFT=UDRIFT
      FFTANG=FTANG
      DO 1045 J=1,NCAB
      DRAG=0.5*RHO*CDC(J)*DC(J)
      START=0.
      WPLA=WCA(J)
      WPLB=WCB(J)
      TREFC=TREF(J)
      JAM=J
      PC=P(J)
      FNP=NPR(J)
      SPA=FLC(J)/FNP
      N1=NLAST+2
      NLAST=N1+NPR(J)-1
      DO 1035 M=N1,NLAST
      MINDEX=M
      Y0(1)=T(M-1)
      Y0(2)=PHI(M-1)
      Y0(3)=X(M-1)
      Y0(4)=Y(M-1)
      Y0(5)=SE(M-1)
      SS=S(M-1)
      CALL KUTMER(5,SS,Y0,EPSLON,SPA,START,HCX,EP2)
      T(M)=Y0(1)
      PHI(M)=Y0(2)
      PHID(M)=PHI(M)*RADIAN
      IF(KIT-1) 1030,1025,1030
1025  IF((PHID(M).GT.125.).AND.(K2.LE.4)) GO TO 1225
      IF((PHID(M).LT.0.).AND.(K2.LE.4)) GO TO 1230
1030  X(M)=Y0(3)
      Y(M)=Y0(4)
      SE(M)=Y0(5)
      S(M)=SS
```

```
1035  CONTINUE
      CALL CUR(Y(NLAST),COFY)
      CREL=COFY-UDRIFT
      IF(NBOD(J).LE.0) GO TO 1040
      CALL BODY(CREL,PHID(NLAST),CDABD(J),NBOD(J),WBD(J),FLIFT(J),
     1BDRAG(J),IRUN,JAM)
1040  DRAGH=0.5*RHO*CDABD(J)*CREL*ABS(CREL)
      XCOMP=DRAGH+T(NLAST)*COS(PHI(NLAST))
      YCOMP=-WBD(J)+T(NLAST)*SIN(PHI(NLAST))
      T(NLAST+1)=SQRT(XCOMP**2+YCOMP**2)
      PHI(NLAST+1)=ATAN2(YCOMP,XCOMP)
      PHID(NLAST+1)=PHI(NLAST+1)*RADIAN
      X(NLAST+1)=X(NLAST)
      Y(NLAST+1)=Y(NLAST)
      S(NLAST+1)=S(NLAST)
      SE(NLAST+1)=SE(NLAST)
1045  CONTINUE
      MPRINT=NLAST
      THORIZ=T(MPRINT)*COS(PHI(MPRINT))
      TVERT=T(MPRINT)*SIN(PHI(MPRINT))
      CALL CUR(Y(MPRINT),COFY)
      I2MANY=0
      CREL=COFY-UDRIFT
      WBOT=WBD(NCAB)
      DRAGBT=0.5*RHO*CDABD(NCAB)*CREL*ABS(CREL)
      IF(ABS(DRAGBT).LT.0.001) DRAGBT=0.001
C ------------------------------------------------------------------
C  CHECK BOTTOM CONDITIONS.
C ------------------------------------------------------------------
      PPERV=PRV
      PPERH=PRH
      ERRORV=TVERT-WBOT-TBV
      ERRORH=THORIZ+DRAGBT-TBH
      PRV=ERRORV
      PRH=ERRORH
      ABSERH=ABS(ERRORH)
      ABSERV=ABS(ERRORV)
      RESULT=SQRT(ERRORH**2+ERRORV**2)
      RATIO1=ABS(ERRORH/DRAGBT)
      RATIO2=ABS(ERRORV/(WBOT+TBV))
      DRGTBH=DRAGBT+TBH
      RATIO3=ABS(ERRORH/DRGTBH)
      IF(RESULT-BRSLT) 1055,1055,1060
1055  BH=H
      BUDR=UDRIFT
      BRSLT=RESULT
1060  IF((RATIO3.LE..02).AND.(RATIO2.LE..02)) GO TO 1240
      IF(ABS(DRGTBH)-0.30) 1065,1065,1070
1065  IF((RATIO3.LE.0.10).AND.(RATIO2.LE..02)) GO TO 1240
1070  CONTINUE
      IRUN=IRUN+1
      INO=IRUN
      UTEMP=UDRIFT
      IF(IB.GT.1) GO TO 1240
      IF(IRUN.GT.50) GO TO 1075
      GO TO 1085
1075  IF(F-ERRORH) 1080,1220,1080
1080  F=ERRORH
1085  IF(INO.GT.400) GO TO 1220
      IF(KIT-1)  1090,1180,1090
```

```
1090  IF(KIT-2) 1100,1095,1095
1095  KUD=KUD+1
      IF((RATIO3.LE.0.02)) GO TO 1115
      IF((KUD.GT.KUSTOP).AND.(RATIO2.GT.0.02)) GO TO 1115
      IF(((PRH/PPERH).GT.1.).AND.(KUD.GE.2)) GO TO 1130
1100  IF((RATIO3.LE.0.02).AND.(KIT.EQ.0)) GO TO 1135
      IF((KUD.GT.KUSTOP).AND.(KIT.EQ.0)) GO TO 1135
1105  IF(ERRORH.GT.0.0) GO TO 1110
      UDRIFT=.5*(UDRIFT+UMIN)
      UMAX=UTEMP
      GO TO 1010
1110  UDRIFT=.5*(UDRIFT+UMAX)
      UMIN=UTEMP
      GO TO 1010
1115  KH=KH+1
      KUD=0
      IF((-ERRORV/PRERV).GT.0.5) DHFAC=0.5*DHFAC
      IF((ERRORV/PRERV).GT.0.7) DHFAC=1.5*DHFAC
      PRERV=ERRORV
      IF((ERRORV.GT.0.0).AND.(PHID(MPRINT).LT.180.)) GO TO 1120
      H=HTEMP+DHFAC*ABSERV/GPRBSQ
      IF(H.GE.HMAXP) H=HTEMP+0.75*(HMAXP-HTEMP)
      HHT=H/HTEMP
      HTH=HTEMP/H
      HMINP=HTEMP
      UMAX=(HHT+.05)*UDRIFT
      UMIN=(HTH-.05)*UDRIFT
      GO TO 1125
1120  H=HTEMP-DHFAC*ABS(ERRORV)/GPRBSQ
      IF(H.LT.(0.5*(HTEMP+HMIN))) H=0.5*(HTEMP+HMIN)
      IF(H.LE.HMINP) H=HTEMP-0.75*(HTEMP-HMINP)
      HMAXP=HTEMP
      HHT=H/HTEMP
      HTH=HTEMP/H
      UMAX=(HTH+.05)*UDRIFT
      UMIN=(HHT-.05)*UDRIFT
1125  PREVH=HTEMP
      PREVU=UDRIFT
      UDRIFT=1.01*UDRIFT
      UMAX1=UMAX
      UMIN1=UMIN
      GO TO 1010
1130  KREV=KREV+1
      IF(KREV.GT.10) GO TO 1220
      GO TO 1105
1135  KIT=1
      UD1=UDRIFT
      H1=H
      EV1=ERRORV
      UMAX=UMAX+ABS(ERRORV)*UDRIFT*.03
      UMIN=UMIN-ABS(ERRORV)*UDRIFT*.03
      IF(WBOT.LT.0.) GO TO 1215
      GO TO 1180
1140  FIRST=-100.
      DO 1145 I=1,MPRINT
      CALL CUR(Y(I),COFY)
      FIRST=100.
      CRELKT(I)=(COFY-UDRIFT)*0.5924
      PHIV(I)=90.0-PHID(I)
1145  CONTINUE
```

```
      FIRST=-100.
      HHALF=.5*HTEMP
      CALL CUR(HHALF,COFY)
      FHALF=(COFY-UDRIFT)*.5924
      FIRST=100.
      TOTV=WBOT+TBV
      TOTH=DRAGBT-TBH
C
      GO TO 1245
1180  K2=0
      IF((IRUN.GT.35).AND.(RESULT.GT.EPRIME).AND.(EPRIME.GT.0.20)) GOTO1
     1215
      IF(RESULT.GT.EPRIME) GO TO 1210
1185  PERV=ERRORV
      PERH=ERRORH
      EPRIME=RESULT
      PUDRFT=UDRIFT
      PH=H
      DELTA=1.
      USEN=1.
1190  DELTAH=-DELTA*ABSERV*ERRORV/(RESULT*GPRBSQ)
      DELTAU=USEN*DELTA*ERRORH*ABSERH/(RESULT*DEN5)
      DELTAU=DELTAU/UDRIFT
      IF(DELTAU) 1195,1195,1200
1195  UMAX=UDRIFT
      UMIN=UDRIFT+DELTAU
      GO TO 1205
1200  UMAX=UDRIFT+DELTAU
      UMIN=UDRIFT
1205  UDRIFT=UDRIFT+DELTAU
      H=H+DELTAH
      HINT=0.7
      IF(H.LT.(HTEMP-HINT*(HTEMP-HMIN))) H=(HTEMP-HINT*(HTEMP-HMIN))
      GO TO 1010
1210  IF((DELTA.LT.0.05).OR.(USEN.GT.500.)) GO TO 1185
      IF((DELTA.LT.0.3).AND.(IRUN.LT.25)) GO TO 1185
      IF(EPRIME.LT.0.1.AND.IRUN.LT.30) GO TO 1185
      EHPH=PRH/PERH
      IF((EHPH.GT.1.).AND.(ABS(PRH).GT.ABS(PRV))) GO TO 1185
      AEHPH=ABS(EHPH)
      ERRORH=PERH
      ERRORV=PERV
      ABSERV=ABS(ERRORV)
      ABSERH=ABS(ERRORH)
      RESULT=EPRIME
      UDRIFT=PUDRFT
      H=PH
      ARVPV=ABS(PRV/PERV)
      DELTA=0.5*DELTA
      USEN=1.
      GO TO 1190
1215  KIT=5
      HTEMP=H1
      UTEMP=UD1
      UDRIFT=UD1
      ERRORV=EV1
      KUD=0
      GO TO 1115
1220  IB=15
      H=BH
```

```
      UDRIFT=BUDR
      GO TO 1010
1225  UMAX=UDRIFT
      I2MANY=I2MANY+1
      MPRINT=MINDEX
      IF(I2MANY.GT.7) GO TO 1235
      UDRIFT=.5*(UDRIFT+UMIN)
      GO TO 1010
1230  UMIN=UDRIFT
      I2MANY=I2MANY+1
      MPRINT=MINDEX
      IF(I2MANY.GT.7) GO TO 1235
      UDRIFT=.5*(UDRIFT+UMAX)
      GO TO 1010
1235  K2=K2+1
      K3=K3+1
      IF(K3.GT.15) GO TO 1215
      I2MANY=0
      UMAX=UMAX+0.03*UDRIFT
      UMIN=UMIN-0.03*UDRIFT
      GO TO 1010
1240  ILAST=10
      GO TO 1140
1245  XX(1)=0.0
      YY(1)=0.0
      SAR(1)=0.0
      BPHI(1)=PHID(MPRINT)
      IF(NHPHS.LE.1) GO TO 1270
      IARRAY=NHPHS
      DO 1250 MK=1,NHPHS
      KK=MK-1
      NMKK=NCAB-KK
1250  IARRAY=IARRAY+NPR(NMKK)
      DO 1255 I=1,IARRAY
      II=MPRINT-I
      BPHI(I+1)=PHID(II)
      XX(I+1)=-X(MPRINT)+X(II)
      YY(I+1)=Y(MPRINT)-Y(II)
      SAR(I+1)=S(MPRINT)-S(II)
1255  CONTINUE
      THETA1=ATAN2(YY(IARRAY+1),XX(IARRAY+1))
      THETAD=90.0-RADIAN*THETA1
      RMAX=0.0
      I32=IARRAY+1
      DO 1260 I=2,I32
      R=SQRT(XX(I)**2+YY(I)**2)
      THETA2=ATAN2(YY(I),XX(I))
      Z=R*ABS(SIN(THETA2-THETA1))
      IF(Z.GT. RMAX)  RMAX=Z
1260  CONTINUE
      DO 1265 I=1,I32
      BPHIV(I)=90.0-BPHI(I)
1265  CONTINUE
1270  CONTINUE
C ---------------------------------------------------------------------
C  Debug format / write statements
C ---------------------------------------------------------------------
C
C
C
```

```
C -----------------------------------------------------------------
C
C
C
C
C -----------------------------------------------------------------
C  Write assumed drift data to "Drift.dat"
C -----------------------------------------------------------------
      IF(IFLIP.EQ.1) UDKNTS=-UDKNTS
      WRITE(4,*) UDKNTS
C -----------------------------------------------------------------
      RETURN
      END
C =================================================================
C
C     Subroutine CUR
C
C   This routine calculates flow for a given depth
C
C =================================================================
      SUBROUTINE CUR(X, FOFX)
C -----------------------------------------------------------------
      COMMON/BLK2/XX(30),YY(30),NPROF,FIRST,RHO,NCUR
C -----------------------------------------------------------------
      IF(FIRST.LT. 0.0) I=1
      IF(X.LT.0.) GO TO 1320
      IF(X.GT.XX(NCUR)) GO TO 1330
      IF((X.GE.XX(I)).AND.(X.LE.XX(I+1))) GO TO 1305
      IF((X.GE.XX(I-1)).AND.(X.LE.XX(I)) ) GO TO 1310
      IF((X.GE.XX(I+1)).AND.(X.LE.XX(I+2))) GO TO 1315
      I=1
1300  IF(X.LE.XX(I+1)) GO TO 1305
      I=I+1
      GO TO 1300
1305  FOFX=YY(I)+((YY(I+1)-YY(I))/(XX(I+1)-XX(I)))*(X-XX(I))
      RETURN
1310  I=I-1
      GO TO 1305
1315  I=I+1
      GO TO 1305
1320  FOFX=YY(1)
      RETURN
1330  FOFX=YY(NCUR)
      END
C =================================================================
C
C     Subroutine DAUX
C
C   This routine sets up the equations of equilibrium to be
C   solved by KUTMER
C
C =================================================================
      SUBROUTINE DAUX(S,IN,DE)
C -----------------------------------------------------------------
      DIMENSION DE(5)
      COMMON /BLK4/ DRAG,WPLA,WPLB,FFTANG,DRIFT,TREFC,PC
      COMMON /BLK7/ FAE(100,15),AE(100,15),NAE(100),JAM
      REAL IN(5)
      CALL CUR(IN(4),COFY)
      IF(NAE(JAM).EQ.1) AEC=AE(JAM,1)
```

```
      IF(NAE(JAM).GT.1) CALL STRETH(IN(1),AEC)
      CREL=COFY-DRIFT
      CABSC=CREL*ABS(CREL)
      E=(IN(1)-TREFC)/AEC
      DE(5)=1.+E
      PCE=(1./(1.+E))**PC
      DRAP=DRAG*PCE
      F2=PCE*PCE*DE(5)
      WPUL=WPLA-WPLB*F2
      DE(3)=-COS(IN(2))*DE(5)
      DE(4)=SIN(IN(2))*DE(5)
      DE(1)=DRAP*CABSC*SIGN(FFTANG,COS(IN(2)))*DE(5)-WPUL*SIN(IN(2))
      DE(2)=-(DRAP*CABSC*SIN(IN(2))*ABS(SIN(IN(2))*DE(5))+WPUL*COS(IN(2)
     1))/IN(1)
      RETURN
      END
C ======================================================================
C
C     Subroutine KUTMER
C
C      KUTMER ROUTINE REVISED FOR IVODE   JAN 30,1964
C      Performs Fourth Order Runge-Kutta integration along cable
C
C ======================================================================
      SUBROUTINE KUTMER(N,T,Y0,EPS,H,FIRST,HCX, A)
C ----------------------------------------------------------------------
      DIMENSION Y0(23),Y1(23),Y2(23),F0(23),F1(23),F2(23)
      IF(FIRST)1505,1500,1505
1500  HC=H
      IPLOC=1
      FIRST=1.
1505  LOC=0
      HCX=HC
1510  CALL DAUX (T,Y0,F0)
      DO 1515 I=1,N
1515  Y1(I)=Y0(I)+(HC/3.)*F0(I)
      CALL DAUX (T+HC/3.,Y1,F1)
      DO 1520 I=1,N
1520  Y1(I)=Y0(I)+(HC/6.)*F0(I)+(HC/6.)*F1(I)
      CALL DAUX (T+HC/3.,Y1,F1)
      DO 1525 I=1,N
1525  Y1(I)=Y0(I)+HC/8.*F0(I)+.375*HC*F1(I)
      CALL DAUX (T+HC/2.,Y1,F2)
      DO 1530 I=1,N
1530  Y1(I)=Y0(I)+HC/2.*F0(I)-1.5*HC*F1(I)+2.*HC*F2(I)
      CALL DAUX (T+HC,Y1,F1)
      DO 1535 I=1,N
1535  Y2(I)=Y0(I)+HC/6.*F0(I)+.66666667*HC*F2(I)+(HC/6.)*F1(I)
      INC=0
      DO 1580 I=1,N
      ZZZ=ABS (Y1(I))-A
      IF(ZZZ) 1540,1545,1545
1540  ERROR = ABS(.2*(Y1(I)-Y2(I)) )
      IF(ERROR-A)1570,1570,1550
1545  ERROR=ABS (.2-.2*Y2(I)/Y1(I))
      IF(ERROR-EPS)1570,1570,1550
1550  CONTINUE
      KYSCIE=12
      CZATR=2.**KYSCIE
      XX= CZATR*ABS(HC)-ABS(H)
```

```
      IF (XX) 1555,1565,1565
1555  CONTINUE
      FIRST = 2.
      RETURN
1565  HC=HC/2.
      IPLOC=2 *IPLOC
      LOC=2 *LOC
      HCX=HC
      GO TO 1510
1570  IF(ERROR*64.-EPS)1580,1580,1575
1575  INC=1
1580  CONTINUE
      T=T+HC
      DO 1585 I=1,N
1585  Y0(I)=Y2(I)
      LOC=LOC+1
      IF(LOC-IPLOC)1590,1610,1610
1590  IF(INC)1610,1595,1610
1595  IF(LOC-(LOC/2)*2)1610,1600,1610
1600  IF(IPLOC-1)1610,1610,1605
1605  HC=2.*HC
      LOC=LOC/2
      IPLOC=IPLOC/2
1610  IF(IPLOC-LOC)1510,1615,1510
1615  RETURN
      END
C ======================================================================
C
C     Subroutine Body
C
C  This routine calculate lift and drag on intermediate bodies
C
C ======================================================================
      SUBROUTINE BODY(V,PHIH,CDA,N,W,LIFT,DRAG,IRUN,J)
C ----------------------------------------------------------------------
      DIMENSION WI(100)
      COMMON /BLK6/ PHIM(10,7),U(10,10),L(10,10,7),D(10,10,7),NBOD(100),
     1NPHI(10),NU(10)
      REAL L,LL,LU,LIFT
C ----------------------------------------------------------------------
      IF(IRUN.GT.1)GO TO 1700
      WI(J)=W
1700  W=WI(J)
      I=0
      K=0
      VA=ABS(V)
      PHIV=90.-PHIH
      PHIMIN=PHIM(N,1)
      PHIMAX=PHIM(N,NPHI(N))
      UMIN=U(N,1)
      UMAX=U(N,NU(N))
C  USE FRESH WATER DENSITY
      RHO=1.94
      IF(PHIV.LE.PHIMIN)GO TO 1720
      K=1
      IF(PHIV.GE.PHIMAX)GO TO 1710
1705  IF((PHIV.GT.PHIM(N,K)).AND.(PHIV.LE.PHIM(N,K+1)))GO TO 1715
      K=K+1
      GO TO 1705
1710  K=NPHI(N)
```

```
      GO TO 1720
1715  RPHI=(PHIV-PHIM(N,K))/(PHIM(N,K+1)-PHIM(N,K))
1720  IF(VA.LE.UMIN)GO TO 1740
      I=1
      IF(VA.GE.UMAX)GO TO 1730
1725  IF((VA.GT.U(N,I)).AND.(VA.LE.U(N,I+1)))GO TO 1735
      I=I+1
      GO TO 1725
1730  I=NU(N)
      GO TO 1740
1735  RV=(VA-U(N,I))/(U(N,I+1)-U(N,I))
1740  IF((K.GE.NPHI(N)).OR.(K.LT.1))GO TO 1745
      IF((I.GE.NU(N)).OR.(I.LT.1))GO TO 1755
C
C       ........BOTH PHIV AND VA ARE WITHIN THE LIFT/DRAG TABLE
C
      LL=RPHI*(L(N,I,K+1)-L(N,I,K))+L(N,I,K)
      LU=RPHI*(L(N,I+1,K+1)-L(N,I+1,K))+L(N,I+1,K)
      LIFT=RV*(LU-LL)+LL
      DL=RPHI*(D(N,I,K+1)-D(N,I,K))+D(N,I,K)
      DU=RPHI*(D(N,I+1,K+1)-D(N,I+1,K))+D(N,I+1,K)
      DRAG=RV*(DU-DL)+DL
      GO TO 1760
1745  IF((I.GE.NU(N)).OR.(I.LT.1))GO TO 1750
C
C  .....VA IS WITHIN THE LIFT/DRAG TABLE, PHIV IS NOT
C
      IF(K.LT.1) K=1
      LIFT=RV*(L(N,I+1,K)-L(N,I,K))+L(N,I,K)
      DRAG=RV*(D(N,I+1,K)-D(N,I,K))+D(N,I,K)
      GO TO 1760
C
C       .....BOTH VA AND PHIV ARE OUTSIDE THE LIFT/DRAG TABLE
C
1750  IF(I.LT.1) I=1
      IF(K.LT.1) K=1
      LIFT=L(N,I,K)
      DRAG=D(N,I,K)
      GO TO 1760
C
C       .....PHIV IS WITHIN THE LIFT/DRAG TABLE, VA IS NOT
C
1755  IF(I.LT.1) I=1
      LIFT=RPHI*(L(N,I,K+1)-L(N,I,K))+L(N,I,K)
      DRAG=RPHI*(D(N,I,K+1)-D(N,I,K))+D(N,I,K)
1760  CDA=2.*DRAG/(RHO*VA**2.)
      W=W-LIFT
      RETURN
      END
C ==================================================================
C
C     Subroutine VAR
C
C   This routine correlates surface unit drag to
C   corresponding flow
C
C ==================================================================
      SUBROUTINE VAR(F1,FLOW,HO,DB,CDB1,CD)
C ------------------------------------------------------------------
      COMMON/BLK8/NFOSB,FOSB(16),VOSB(16)
```

```
C -----------------------------------------------------------------
C Use fresh water density
C -----------------------------------------------------------------
      RHO=1.94
      N=NFOSB
C -----------------------------------------------------------------
C Convert from ft/s to kts
C -----------------------------------------------------------------
      FLOWK=.5924*FLOW
C -----------------------------------------------------------------
C Flow / drag table interpolation
C -----------------------------------------------------------------
      IF(FLOWK.LT.VOSB(1)) GO TO 1810
      IF(FLOWK.GT.VOSB(N)) GO TO 1815
      I=1
1800  IF(FLOWK.LE.VOSB(I+1)) GO TO 1805
      I=I+1
      GO TO 1800
1805  F2=FOSB(I)+(FOSB(I+1)-FOSB(I))/(VOSB(I+1)-VOSB(I))*
     1(FLOWK-VOSB(I))
      GO TO 1820
1810  F2=FOSB(1)
      GO TO 1820
1815  F2=FOSB(N)
      GO TO 1820
C -----------------------------------------------------------------
C  Calculate CD
C -----------------------------------------------------------------
1820  CD1=2*F2/(RHO*FLOW*FLOW*DB*HO)
C -----------------------------------------------------------------
C  Compare interpolated drag to drag calculated in STEADY
C -----------------------------------------------------------------
      EDRG=F2-F1
      IF(ABS(EDRG).LE..001) GO TO 1835
      IF(F2-F1) 1825,1825,1830
1825  CD=CDB1-ABS((CD1-CDB1)/2)
      GO TO 1840
1830  CD=CDB1+ABS((CD1-CDB1)/2)
      GO TO 1840
1835  CD=CDB1
1840  RETURN
      END
C =================================================================
C
C     Subroutine STRETH
C
C   This routine calculates AE at given force values
C
C =================================================================
      SUBROUTINE STRETH(X,AEC)
C -----------------------------------------------------------------
      COMMON /BLK7/ FAE(100,15),AE(100,15),NAE(100),JAM
C -----------------------------------------------------------------
      J=JAM
      N=NAE(J)
      IF(X.LE.FAE(J,1)) GO TO 1910
      IF(X.GE.FAE(J,N)) GO TO 1915
      I=1
1900  IF(X.LE.FAE(J,I+1)) GO TO 1905
      I=I+1
```

```
      GO TO 1900
1905  AEC=AE(J,I)+(AE(J,I+1)-AE(J,I))/(FAE(J,I+1)-FAE(J,I))*(X-FAE(J,I))
      RETURN
1910  AEC=AE(J,I)
      RETURN
1915  AEC=AE(J,N)
      RETURN
      END
C ======================================================================
C ======================================================================
C ======================================================================
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B
SONOBUOY FIELD DRIFT MODEL MATLAB SCRIPT LISTING

```
function SFDMv2
% -------------Sonobuoy Field Drift Model ver 2.0 ---------------------
%
%   This m-file runs the Sonobuoy Field Drift Model (SFDM)
%
%   Prior to running the main program create set up file using
%   "Drift Model Set Up.xls" as a template. Save the set up file
%   under a different name -- the "Simulation Name".
%   Make sure the correct environmental filevare stored in the
%   Environmental Database folder.
%
%   User m-functions called:
%       saveDMCF, cp_extract, cp_ convert, ff_write, read_drift,
%       update_posit, show_status
%
%   Created by: Dave Hammond
%   Created on: 10-20-2004
%
%   Modifications:
%       ver 1.1 / Dave Hammond / 10-30-2004
%           - Compatible with FF2E_D10
%           - Added plotPath post processing routine
%           - Added 'idBuoy' data to 'initBuoy.mat' file and
%             set up files
%
%       ver 1.2 / Dave Hammond / 11-02-2004
%           - u, v data added to output file
%           - output file renamed to buoyOutData
%           - added current profile output data in curOutData
%
%       ver 1.3 / Dave Hammond / 11-05-2004
%           - updated FF2E call to version FF2E_D11 (fix NDP Exception errors
%             caused by negative relative flows).
%           - name output data after run name
%
%       ver 2.0 / Dave Hammmond / 11-06-2004
%           - consolidate subroutines into one main file (SFDMv2)
%           - clean up code / standardize comment lines
%           - make transportable to other computers (no need to change path
              statements)
%           - add environmental data base folder
%
%       ver 2.5 / Dave Hammond / 11-22-2004
%           - made changes to cp_convert and read_drift to account for
%             drift errors that occur due to using the N/E global coordinate
%             system. cp_convert calculates a "dominate" current direction
%             based on a weighted mean of current energy, and transforms the
%             u and v current profiles into a new orthogonal coordinate system
%             aligned with the dominate current direction to pass along to
%             FF2E_D11. read_drift transforms the u and v drift back to the
%             global N/E coordinate system
%
% ------------------------------------------------------------------
```

```
clear all
% Declare global variables
global dName pName
%
% Display program title
head
%
fprintf('     Note: Make sure you are in the "Drift Model" folder before
running\n\n');
fprintf('   Current path is:\n     %s',cd);
fprintf('\n\n');
pChange = input('   Continue? (y / n)  ','s');
if lower(pChange) == 'n'
    clc
    return
end
clc
%
% -------------------------------------------------------
% Get Input and Control Data
% -------------------------------------------------------
% Display program title
head
% Name of the Excel input file
fName = input('\n   Enter Simulation Name: ', 's');
% Directory name that contains the simulation files
dName = input('\n   Enter Simulation Folder Name: ', 's');
% Set path for simulation
pName = cd;
addpath(pName);
cd([pName '\' dName]);
% Retrieve control parameters from set up file and load variables
save_DMCF(fName);
load 'TimeInit';
load 'BuoyInit';
%
% -------------------------------------------------------
% Main Routine -- calculate buoy trajectories
% -------------------------------------------------------
% Buoy Loop
for i = 1:nBuoy
    % Set time vector for buoy i
    t = [t0Buoy(i):tStep:tStop]';
    nTime = size(t,1);
    % Initialize buoy position and velocity variables
    x(1) = x0Buoy(i);
    y(1) = y0Buoy(i);
    u(1) = 0;
    v(1) = 0;
    % Time Loop
    for j = 1:(nTime-1)
        % Extract current profiles at buoy position (xBuoy, yBuoy)
        % and time (tBuoy)
        [uCur, zCur]=cp_extract(t(j),x(j),y(j));
        % convert current profile for FF2E
        [uKts, zFt, uDir] = cp_convert(uCur, zCur);
        % write ff2e file using buoy data and extracted current profile
        ff_write('IN.DAT', typeBuoy{i,1});
        % run ff2e
        cd(pName);
```

```
        !FF2E_D11
        cd([pName '\' dName]);
        % read drift data from FF2E results and transform to N/E coordinates
        [u(j+1), v(j+1)] = read_drift('DRIFT.DAT', uDir);
        % calculate new position
        [x(j+1), y(j+1)] = update_posit(u(j+1), v(j+1), x(j), y(j), tStep);
        % Display status during run
        show_status(x(j), y(j), u(j+1), v(j+1), t(j+1), i);
  end
    % Store output data in 'outData' variable
    buoyOutData(i) = {[x', y', t, u', v']};
    curOutData(i) = {[uCur',zCur]};
    clear x y t u v
end
% Save output data and delete FF2E .dat files
save(fName,'buoyOutData')
cd ..
dos('del in.dat');
dos('del drift.dat');
% ==================================================================
%
%                           Subroutines
%
% ==================================================================
%       save_DMCF
% ==================================================================
function save_DMCF(fName)
% ------------------------------------------------------------------
%  This function reads the drift model  control parameters from
%  the user interface file: 'fName'
%  and saves them as variables in
%  four MAT-files:
%   BuoyInit
%       nBuoy       number of buoys in field
%       typeBuoy    buoy types / depths
%       x0Buoy      initial buoy longitude (dd.ddd)
%       y0Buoy      initial buoy latitude (dd.ddd)
%       t0Buoy      buoy start times (serial days)
%   curInit
%       curDB       current database filename
%       curData     grid start position (long, lat), spacing (deg),
%                   data start time (serial day) [x0, y0, s, t0]
%   windInit
%       windDB      wind database filename
%       windData    x0, y0, s, t0
%   timeInit
%       tStart      simulation start time (serial day)
%       tStop       simulation stop time (serial day)
%       tStep       simulation time step (day)
%
%
%  fName    Drift model simulation filename (based on Excel file
%           "Drift Model Setup")
%
%  User m functions called: none
%
%  Created by: Dave Hammond, NAWC AD 4.5.14.2
%  Created on: 10-13-2004
%
% ---------------------------------------------------------------------
```

```
% Declare global variables
global dName pName
%
% Constants
tConvert = datenum('12/30/1899');        % time correction for Excel
%
% Get deployment data
[data, names] = xlsread(fName,'Deployment');
nBuoy = size(data,1);
for i = 1:nBuoy
    typeBuoy(i,:) = names(i+1,1) ;
end
idBuoy = data(:,1);
x0Buoy = data(:,2);
y0Buoy = data(:,3);
t0Buoy = tConvert + data(:,4) + data(:,5);
save 'BuoyInit' nBuoy typeBuoy idBuoy x0Buoy y0Buoy t0Buoy
%
% Get Environmental Data
% Current
[data, names] = xlsread(fName,'Environment');
curDB = names{2,2};
curGrid = data(1:5,2);
curTime(1) = tConvert + data(6,2) + data(7,2);
curTime(2) = tConvert + data(8,2) + data(9,2);
curTime(3) = data(10,2);
curDepth = data(11:size(data,1),2);
% Wind
windDB = names{3,2};
windGrid = data(1:5,3);
windTime(1) = tConvert + data(6,3) + data(7,3);
windTime(2) = tConvert + data(8,3) + data(9,3);
windTime(3) = data(10,3);
save 'CurInit' curDB curGrid curTime curDepth
save 'WindInit' windDB windGrid windTime
%
% Get Time Data
[data, names] = xlsread(fName,'Time');
tStart = tConvert + data(1) + data(2);
tStop = tConvert + data(3) + data(4);
tStep = data(5);
save 'TimeInit' tStart tStop tStep
% ==================================================================
%       cp_extract
% ==================================================================
function [uCur, zCur]=cp_extract(tBuoy,xBuoy,yBuoy);
% ------------------------------------------------------------------
%   function to extract a current profile at buoy location
%     x, y and time, t from gridded CUPOM GOM model data
% ------------------------------------------------------------------
%
%  input variables:
%     tBuoy:  time for requested profile (days)
%     xBuoy:  buoy x position (dec deg long)
%     yBuoy:  buoy y position (dec deg lat)
%
%  output variables:
%     uCur:  current velocity vector
%     zCur:  corresponding depth (m)
%
```

```
% ----------------------------------------------------------------
% Declare global variables
global dName pName
%
% load current profile and time parameters
load curInit;
load timeInit;
% load current data from environmental database folder
curData = [pName '\Environmental Database\' curDB];
load(curData);
x0Cur = curGrid(1);
y0Cur = curGrid(2);
xfCur = curGrid(3);
yfCur = curGrid(4);
sCur = curGrid(5);
t0Cur = curTime(1);
tfCur = curTime(2);
dtCur = curTime(3);
zCur = curDepth;
nLayer = size(zCur,1);
%  make current grid
x = [x0Cur:sCur:xfCur];
y = [y0Cur:sCur:yfCur];
t =[t0Cur:dtCur:tfCur];
[X,Y,T]=ndgrid(x,y,t);
% calculate uBuoy and vBuoy for each layer using interpn function
for i = 1:nLayer
  ustr=['u' num2str(i)];
  vstr=['v' num2str(i)];
  eval(['u=' ustr ';']);
  eval(['v=' vstr ';']);
  uCur(1,i) = interpn(X, Y, T, u, xBuoy, yBuoy, tBuoy);
  uCur(2,i) = interpn(X, Y, T, v, xBuoy, yBuoy, tBuoy);
end
% ================================================================
%       cp_convert
% ================================================================
function [uKts, zFt, uDir] = cp_convert(uCur, zCur)
% -----------------------------------------------------------------
% [uKts, zFt] = cp_convert(uCur, zCur)
%
%  This function calculates the "dominant" current direction (uDir)
%  and transforms uCur onto a new coordinate system along uDir
%
%  This function converts the extracted current profiles into knots
%  and the depth to feet. It also reverses the current profile
%  direction if the average current is negative (this will cause
%  FF2E to fail)
%
%  Input Variables
%   uCur(1,:)            u direction current profile (m/s)
%   uCur(2,:)            v direction current profile (m/s)
%   zCur                 depth vector (m)
%
%  Output Variables
%   uKts                 u & v direction current profile (kts)
%   zFt                  depth vector (ft)
%   uDir                 dominant current profile direction
%
%  User m functions called: rot_mat
```

```
%  .MAT files created: curProfile.mat
%
%  Created by: Dave Hammond, NAWC AD 4.5.14.2
%  Created on: 10-19-2004
%
%  Modification History:
%   1. Removed reverse current functions for compatibility with
%       FF2E_D10. (10/30/04 by DSH)
%
%   2. Added dominant current transformation (11/22/04 by DSH)
%
% ---------------------------------------------------------------------
%
% Calculate dominante current direction using a weighted average
% weighting based on "current energy" u^2
phi = atan2(uCur(2,:), uCur(1,:));
Q = uCur(1,:).^2 + uCur(2,:).^2;
q = Q ./ sum(Q);
uDir = sum(q .* phi);
%
% Assign rotation matrix
A = rot_mat(uDir);
%
% Transform u / v currents from the global N/E x-y axis to the new
% axis x'-y' aligned with uDir
uDom = A * uCur;
%  Convert to knots and feet
uKts(1,:) = 1.9438 * uDom(1,:);
uKts(2,:) = 1.9438 * uDom(2,:);
zFt = 3.281 * zCur;
%  Give uKts(2,:) some small current if = 0 to avoid FF2E errors
if mean(uKts(2,:)) < .001
    uKts(2,1) = .01;
end
% save variables in file for ffwrite
save 'curProfile' uKts zFt
% ==================================================================
%       ff_write
% ==================================================================
function ff_write(FF2Ename, buoyData);
%
% This function reads the FF2E input data from a .MAT file
%   and writes a formatted FF2E input file for use with
%
%   FF2Ename = FF2E input file name
%   buoyData = .MAT file containing sonobuoy FF2E input variables
%   curProfile = .MAT file containing current profile data
%
%   Created by: Dave Hammond, NAWC AD 4.5.14.2
%   Created on: 10-13-2004
%
% -----------------------------------------------------------
% Declare global variables
global dName pName
%%
% Open input file to write and load buoy / current variables
%
pName2 = [pName '\Sonobuoy Database\'];
fid = fopen([pName '\' FF2Ename],'wt');
load([pName2 buoyData]);
```

```
load('curProfile');
NCUR = size(zFt);
%
% Format statements
%
f2 = '%10.6f%10.6f%10.6f%10.6f%10.6f%10.6f%10.6f%10.6f\n';
f3 = '%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f\n';
f4 = '%12.4f%12.6f%12.6f%12.6f%12.6f%3i';
f5 = '%12.6f%12.6f%12.4f%12.6f%3i';
f8 = '%4i';
%
% Write fomatted data
%
% Control and surface float parameters
NCAB = NCAB(1);
NTAB = NTAB(1);
NFOSB = NFOSB(1);
for k = 1:NCASES(1)
    fprintf(fid,f8,NCUR(1), NCAB, NTAB, NFOSB(1), NHPHS(1), NCTR(1));
    fprintf(fid,'\n');
    fprintf(fid,f2,DIA(1), LA(1), CDA(1), TBH(1), TBV(1));
    fprintf(fid,'\n');
    fprintf(fid,f2, DB(1), LB(1), CDB1(1), CDB2(1), WB(1), UWINDK(1));
    fprintf(fid,'\n');
% Surface float drag data
    if NFOSB ~=0
        fprintf(fid, f3, FOSB(1:NFOSB));
        if rem(NFOSB,8) > 0
            fprintf(fid,'\n');
        end
        fprintf(fid, f3, VOSB(1:NFOSB));
        if rem(NFOSB,8) > 0
            fprintf(fid,'\n');
        end
    end
    fprintf(fid,f2, CDAPK(1), WPAK(1), RHO(1));
    fprintf(fid,'\n');
% Cable Data
    for i = 1:NCAB
        fprintf(fid,f4, FLC(i), DCI(i), WC(i), CDC(i), CVFAC(i), NPR(i));
        fprintf(fid,'\n');
    end
    fprintf(fid,f8,NAE(1:NCAB));
    fprintf(fid,'\n');
% Cable Elasticity
    for i = 1:NCAB
        fprintf(fid,f3,AE(i,1:NAE(i)));
        if rem(NAE(i),8) > 0
            fprintf(fid,'\n');
        end
        if NAE(i) ~=1
            fprintf(fid, f3, FAE(i,1:NAE(i)));
            if rem(NAE(i),8) > 0
                fprintf(fid,'\n');
            end
        end
    end
% Body Data
    for i = 1:NCAB
        fprintf(fid,f5, WBD(i), CDABD(i), TREF(i), P(i), NBOD(i));
```

```
        fprintf(fid,'\n');
    end
% Current Profile Data
    fprintf(fid,'%10.2f',zFt);
    fprintf(fid,'\n');
    fprintf(fid,'%10.6f',uKts(k,:));
    fprintf(fid,'\n');
% Lift Drag Tables
    if NTAB ~= 0
        for i = 1:NTAB
            fprintf(fid, f8, NPHI(i), NU(i));
            fprintf(fid,'\n');
            fprintf(fid, f3, PHIM(i,1:NPHI(i)));
            if rem(NPHI(i),8) > 0
                fprintf(fid,'\n');
            end
            fprintf(fid, f3, U(i,1:NU(i)));
            if rem(NU(i),8) > 0
                fprintf(fid,'\n');
            end
            N = NPHI(i) * NU(i);
            fprintf(fid, f2, DDRAG(i, 1:N));
            if rem(N,8) > 0
                fprintf(fid,'\n');
            end
            fprintf(fid, f2, DLIFT(i, 1:N));
            if rem(N,8) > 0
                fprintf(fid,'\n');
            end
        end
    end
end
fclose(fid);
% ==================================================================
%        read_drift
% ==================================================================
function [u, v] = read_drift(fname, uDir)
% ------------------------------------------------------------------
%  This function reads the calculated drift speed from the FF2E
%  output file named 'fname'. Transforms u & v from uDir coordinate
%  system back to N/E system
%
%  Input Variables:
%   fname                 FF2E output file name
%   uDir                  "Dominant" current direction
%
%  Output Variables
%   u, v                  u & v direction current (m/s)
%
%  User m functions called: none
%  .MAT files created: none
%
%  Created by: Dave Hammond, NAWC AD 4.5.14.2
%  Created on: 10-19-2004
%
%  Modification History:
%   1. Removed reverse current functions for compatibility with
%       FF2E_D10. (10/30/04 by dsh)
%   2. Read drift data from "DRIFT.DAT" created by FF2E_D10
%       (10/30/04 by dsh)
```

```
%   3. Added axis rotation from dominant current axis back to N/E
%       global axis
%
% ---------------------------------------------------------------------
% Declare global variables
global dName pName
%
% load drift output file into variable 'uD'
uD = load('-ascii',[pName '\' fname]);
% transform to N/E coordinate system
A = rot_mat(uDir);
uN = inv(A) * uD;
% Convert from kts to m/s
uN = uN * .5144;
u = uN(1); v = uN(2);
% ==================================================================
%       update_posit
% ==================================================================
function [xNew, yNew] = update_posit(u, v, x, y, dt)
% ---------------------------------------------------------------------
%  This function calculates a new buoy position based on the drift
%  velocity, time step and current postion
%
%  Input Variables
%   u, v                    u & v drift velocity (m/s)
%   x, y                    current x and y position (lat/long)
%   dt                      time step
%
%  Output Variables
%   xNew, yNew              updated buoy postion (lat/long)
%
%  User m functions called: none
%  .MAT files created: none
%
%  Created by: Dave Hammond, NAWC AD 4.5.14.2
%  Created on: 10-19-2004
%
% ---------------------------------------------------------------------
%
% convert drift from m / s to deg / day
    deg2rad = pi/180.0d0 ;
    vscale = (8.64d4/1.0d3)*(180.0d0/(pi*6371.0d0)) ;
    uscale = vscale*cos(deg2rad*y) ;
    u = u * uscale;
    v = v * vscale;
% calculate new buoy position
    xNew = x + u * dt;
    yNew = y + v * dt;
%
% ==================================================================
%       Show Status
% ==================================================================
function show_status(x, y, u, v, t, i)
%
%  Function to display run and buoy status during run
%   Simulation time displayed
%   Buoy position and ID displayed
%   Drift velocity and heading displayed
%
% ------------------------------------------------------------------
```

```
%   Input variables
%       x, y:   buoy position
%       u, v:   buoy velocity
%       t:      simulation time
%       i:      buoy id
%   Output variables
%       none
%
% -------------------------------------------------------------
%   Dave Hammond, NAWCAD 4.5.14.2
%   11.02.04
% -------------------------------------------------------------
%
% calculate buoy drift (m/s) and heading (deg)
drift = sqrt(u^2 + v^2);
heading = atan2(u, v) * (180 / pi);
if heading < 0
    heading = 360 + heading;
end
% Display program title
head
%
fprintf('                Buoy #%i \n\n', i);
fprintf('         Time = %s \n', datestr(t,0));
fprintf('         Drift Speed = %6.2f m/s \n',drift);
fprintf('         Heading = %6.2f deg \n',heading);
fprintf('         x = %6.2f deg long\n',x);
fprintf('         y = %6.2f deg lat \n\n',y);
fprintf('     *************************************************\n\n\n');
% =============================================================
%
% =============================================================
function head
% ----------------------------------------------------------------
clc
fprintf('\n\n\n');
fprintf('     *************************************************\n\n');
fprintf('                        Sonobuoy Field Drift Model \n');
fprintf('                              SFDM ver 2.0\n\n');
fprintf('     *************************************************\n\n');
%
% =============================================================
%       Rotation Matrix
% =============================================================
function A = rot_mat(phi)
%
%  This function sets up the rotation matrix for a
% clockwise rotation, phi, of the coordinate axis
%
% -------------------------------------------------------------
%   Input variables
%       phi:    angle of rotation
%   Output variables
%       A:      rotation matrix
%
% -------------------------------------------------------------
%   Dave Hammond, NAWCAD 4.5.14.2
%   11.22.04
% -------------------------------------------------------------
%
```

```
%
A(1,1) = cos(phi);
A(1,2) = sin(phi);
A(2,1) = -sin(phi);
A(2,2) = cos(phi);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C
SONOBUOY FIELD DRIFT MODEL POST MATLAB SCRIPT LISTING

```
function SFDMpost()
% -------------SFDM Post Processing Routines ver 1.0 ---------------------
%
%   This m-file post processes data from the Sonobuoy Field Drift Model (SFDM)
%
%   This file will process data from the last SFDM run conducted
%
%   User m-functions called:
%
%   Created by: Dave Hammond
%   Created on: 11-01-04
%
%   Modifications:
%
%   ver 0.2 / Dave Hammond / 11-07-04
%        - added background picture (map) for trajectory plots
%        - added movie creation selection -- save to .avi file
%
%   ver 1.0 / Dave Hammond / 11-08-04
%        - finished backgournd picture capability
%        - refined markers
%
% -----------------------------------------------------------------
% Global variables
global fName
% Display program title
head
fprintf('    Note:   Make sure you are in the "Drift Model" folder before
running\n\n');
fprintf('   Current path is:\n     %s',cd);
fprintf('\n\n');
pChange = input('   Continue? (y / n)   ','s');
if lower(pChange) == 'n'
    clc
    return
end
% Name of the output data file
fprintf('\n');
fName = input('   Enter Simulation Name: ', 's');
fprintf('\n');
% Directory name that contains the simulation files
dName = input('   Enter Simulation Folder Name: ', 's');
% Set path for simulation
pName = cd;
addpath(pName);
cd([pName '\' dName]);
iPlot = 1;
while(iPlot > 0)
    % Display program title
    head
    % Display plot options
    fprintf('       Plot Options \n\n');
    fprintf(' 1 = Plot buoy trajectories\n');
    fprintf(' 2 = Plot buoy velocities\n');
    fprintf(' 3 = Plot buoy headings\n');
```

```
    fprintf(' 4 = Make movie\n');
    fprintf(' 5 = Not implemented\n');
    fprintf(' 6 = Not implemented\n');
    fprintf(' 7 = Not implemented\n');
    fprintf(' 8 = Not implemented\n');
    fprintf('\n 0 = Exit program\n');
    fprintf('\n=================================\n\n')
    iPlot = input('Enter plot option: ');
    switch iPlot
    case 1
        plot_path
    case 2
        plot_vel
    case 3
        plot_head
    case 4
        mov_path
    case 5
        clc
        fprintf('\n\n Not implemented yet. Press any key to continue...');
        pause
    case 6
        clc
        fprintf('\n\n Not implemented yet. Press any key to continue...');
        pause
    case 7
        clc
        fprintf('\n\n Not implemented yet. Press any key to continue...');
        pause
    case 8
        clc
        fprintf('\n\n Not implemented yet. Press any key to continue...');
        pause
    case 0
        clc
        fprintf('\n\n  Have a nice day........\n\n');
        cd ..
        break
    end
end
% ================================================================
%
%    Subroutines
%
% ================================================================
%
% ================================================================
function head
%
clc
fprintf('\n\n\n');
fprintf('     ******************************************************\n\n');
fprintf('                         Sonobuoy Field Drift Model \n');
fprintf('                          Post Processing Routine \n');
fprintf('                              SFDMpost ver 1.0 \n\n');
fprintf('     ******************************************************\n\n\n');
% ================================================================
%
% ================================================================
function plot_path
```

```
% ------------------------------------------------------------------
global fName
%
load BuoyInit
load CurInit
load(fName)
% Display program title
head
% Image file name
iName = '';
iName = input('  Image filename for plot background (press Enter for no file):
','s');
fprintf('\n');
% Number of buoys to plot
iBuoy = input('  Number of buoy to plot (0 = all): ');
fprintf('\n');
% set up buoy plot limits
plotLim = [-88 -80 23 28];
if exist('plotLim') == 1
    fprintf('  Current image limits are:\n')
    fprintf('                                   %5.2f to %5.2f deg Longitude\n',
plotLim(1), plotLim(2));
    fprintf('                                    %5.2f to %5.2f deg Latitude\n',
plotLim(3), plotLim(4));
    pQ = input('  Change image plot limits (y / n)? ','s');
    if lower(pQ) == 'y'
        plotLim = input('  Enter new image plot limits [xMin xMax yMin yMax]:
');
    end
else
    plotLim = input('  Enter image plot limits [xMin xMax yMin yMax]: ');
end
figure
axis(plotLim);
if iName ~= ''
    imData = imread(iName);
    xa = [plotLim(1) plotLim(2)];
    ya = [plotLim(3) plotLim(4)];
    image(xa, ya, imData);
    grid on
end
% Plot all buoys
grid on
hold on
ha = gca;
set(ha,'YDir', 'normal');
if iBuoy == 0
    for i = 1:nBuoy
        x = buoyOutData{1,i}(:,1);
        y = buoyOutData{1,i}(:,2);
        t = buoyOutData{1,i}(:,3);
        N = size(x,1);
        plot(x,y,'b:')
        plot(x(1),y(1),'bo', 'MarkerSize', 4)
        plot(x(N), y(N), 'k.')
    end
% Plot selected buoys
else
    for j = 1:iBuoy
        fprintf('\n Enter #%i Buoy ID to plot:  ',j);
```

```
        i = input(' ');
        fprintf('\n\n');
        x = buoyOutData{1,i}(:,1);
        y = buoyOutData{1,i}(:,2);
        t = buoyOutData{1,i}(:,3);
        N = size(x,1);
        plot(x,y,'b:')
        plot(x(1),y(1),'bo', 'MarkerSize', 4)
        plot(x(N), y(N), 'k.')
    end
end
xlabel('Longitude (deg)');
ylabel('Latitude (deg)');
title([fName ' Trajectories']);
% ==================================================================
%
% ==================================================================
function plot_vel
% ------------------------------------------------------------------
global fName
%
load BuoyInit
load CurInit
load(fName)
% Display program title
head
% Number of buoys to plot
nBuoy = input('  Number of buoys to plot (0 = all): ');
figure
hold on
% Plot all buoys
if nBuoy == 0
    nBuoy = size(buoyOutData,2);
    for i = 1:nBuoy
        u = buoyOutData{1,i}(:,4);
        v = buoyOutData{1,i}(:,5);
        t = buoyOutData{1,i}(:,3);
        time = t(:) - t(1);
        drift = sqrt(u.^2 + v.^2);
        N = size(u,1);
        plot(time(2:N),drift(2:N))
    end
% Plot selected buoys
else
    for j = 1:nBuoy
        fprintf('\n Enter #%i Buoy ID to plot:  ',j);
        i = input(' ');
        u = buoyOutData{1,i}(:,4);
        v = buoyOutData{1,i}(:,5);
        t = buoyOutData{1,i}(:,3);
        time = t(:) - t(1);
        drift = sqrt(u.^2 + v.^2);
        N = size(u,1);
        plot(time(2:N),drift(2:N))
    end
end
grid
axis([0 time(N) 0 3]);
xlabel('Time (days)');
ylabel('Drift Speed (m/s)');
```

```
title([fName ' Drift Speed']);
% ===================================================================
%
% ===================================================================
function plot_head
% -------------------------------------------------------------------
% Global variables
global fName
%
load BuoyInit
load CurInit
load(fName)
% Display program title
head
% Number of buoys to plot
nBuoy = input('  Number of buoys to plot (0 = all): ');
figure
hold on
% Plot all buoys
if nBuoy == 0
    nBuoy = size(buoyOutData,2);
    for i = 1:nBuoy
        u = buoyOutData{1,i}(:,4);
        v = buoyOutData{1,i}(:,5);
        t = buoyOutData{1,i}(:,3);
        time = t(:) - t(1);
        heading = atan2(u, v) .* (180 / pi);
        N = size(u,1);
        plot(time(2:N),heading(2:N))
    end
% Plot selected buoys
else
    for j = 1:nBuoy
        fprintf('\n Enter #%i Buoy ID to plot:  ',j);
        i = input(' ');
        u = buoyOutData{1,i}(:,4);
        v = buoyOutData{1,i}(:,5);
        t = buoyOutData{1,i}(:,3);
        time = t(:) - t(1);
        heading = atan2(u, v) .* (180 / pi);
        N = size(u,1);
        plot(time(2:N),heading(2:N))
    end
end
grid
axis([0 time(N) -180 180]);
xlabel('Time (days)');
ylabel('Heading (deg)');
title([fName ' Heading']);
% ===================================================================
%
% ===================================================================
function mov_path
% -------------------------------------------------------------------
global fName
%
load BuoyInit
load CurInit
load TimeInit
load(fName)
```

```
% Display program title
head
% Image file name
iName = '';
iName = input('  Image filename for plot background (press Enter for no file):
','s');
fprintf('\n');
% Number of buoys to plot
numBuoy = input('  Number of buoy to plot (0 = all): ');
plotLim = [-88 -80 23 28];
if exist('plotLim') == 1
    fprintf('  Current image limits are:\n')
    fprintf('                                %5.2f to %5.2f deg Longitude\n',
plotLim(1), plotLim(2));
    fprintf('                                %5.2f to %5.2f deg Latitude\n',
plotLim(3), plotLim(4));
    pQ = input('  Change limits (y / n)? ','s');
    if lower(pQ) == 'y'
        plotLim = input('  Enter new image plot limits [xMin xMax yMin yMax]:
');
    end
else
    plotLim = input('  Enter image plot limits [xMin xMax yMin yMax]: ');
end
% Creat time vector
time = [tStart:tStep:tStop]';
nTime = size(time,1);
figure
for j = 1:nTime
    % Set up figure / display background image if available
    axis(plotLim);
    if iName ~= ''
        imData = imread(iName);
        xa = [plotLim(1) plotLim(2)];
        ya = [plotLim(3) plotLim(4)];
        image(xa, ya, imData);
    end
    grid on
    hold on
    ha = gca;
    set(ha,'YDir', 'normal');
    % Plot all buoys
    if numBuoy == 0
        nBuoy = size(buoyOutData,2);
        for i = 1:nBuoy
            x = buoyOutData{1,i}(:,1);
            y = buoyOutData{1,i}(:,2);
            t = buoyOutData{1,i}(:,3);
            if j == 1
                k = find(t <= time(j));
            else
                k = find(t <= time(j) & t > time(j-1));
            end
            if exist('k') == 1
                plot(x(1), y(1), 'bo', 'MarkerSize', 3)
                hold on
                plot(x(1:k), y(1:k), 'b:')
                plot(x(k), y(k), 'k.')
            end
        end
```

```
        % Plot selected buoys
    else
        nBuoy = numBuoy;
        for j = 1:nBuoy
                fprintf('\n Enter #%i Buoy ID to plot:  ',j);
                    i = input(' ');
                fprintf('\n\n');
            end
    end
    axis([curGrid(1) curGrid(3) curGrid(2) curGrid(4)]);
    grid on
    xlabel('Longitude (deg)');
    ylabel('Latitude (deg)');
    title([fName ' Trajectories']);
    tText = datestr(time(j));
    text(curGrid(1)+.125, curGrid(4)-.125,tText);
    hold off
    M(j) = getframe(gcf);
end
%figure
%movie(M);
movie2avi(M, fName, 'fps', 1, 'quality', 100);
% =================================================================
%
% =================================================================
%
% -----------------------------------------------------------------
%
% -----------------------------------------------------------------
%
%
% -----------------------------------------------------------------
%
%
% -----------------------------------------------------------------
%
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D
INDIVIDUAL BUOY SONOBUOY FIELD DRIFT MODEL PLOTS

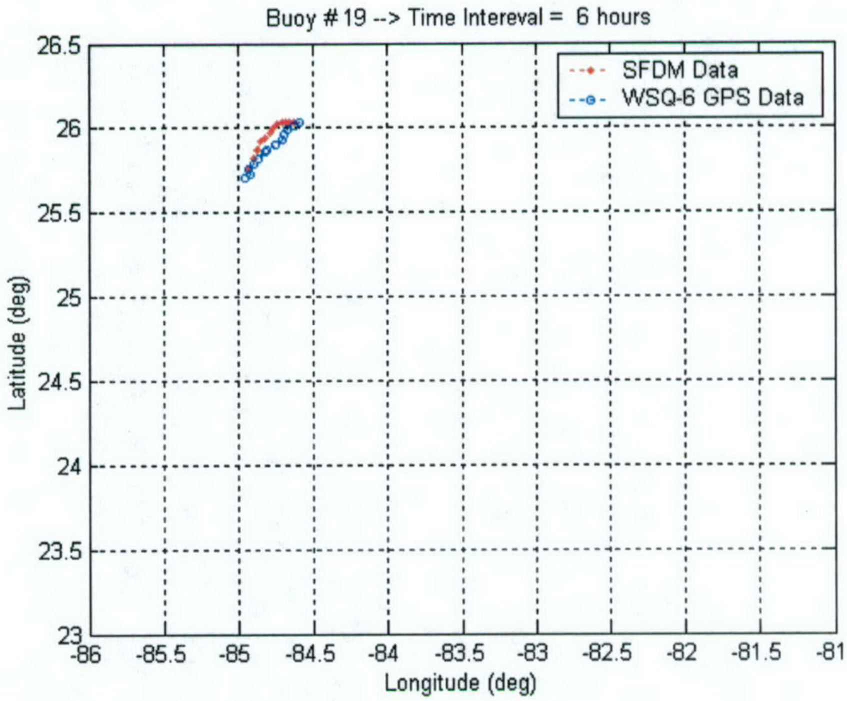


Figure D-1: Buoy 19 – Typical NW Region Trajectory

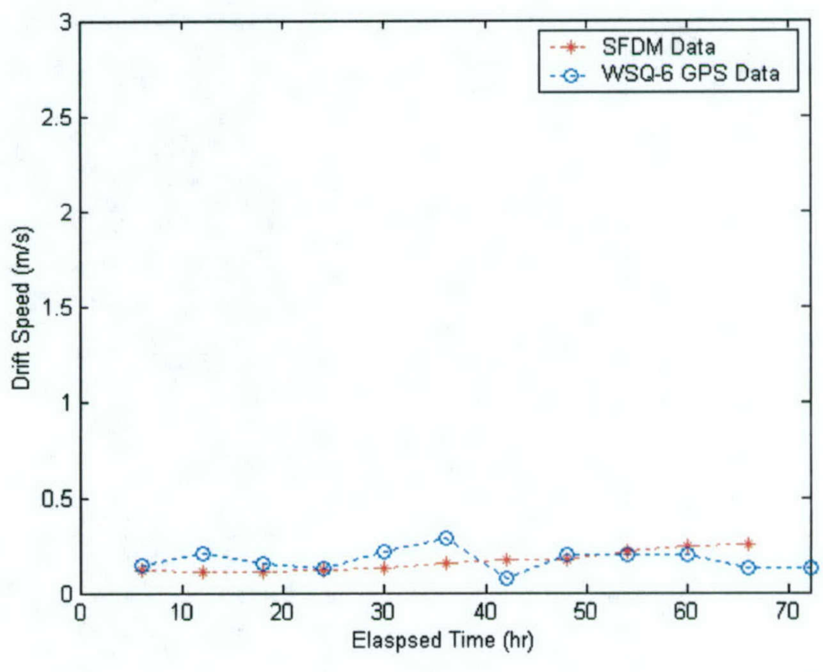


Figure D-2: Buoy 19 – Typical NW Region Velocity History

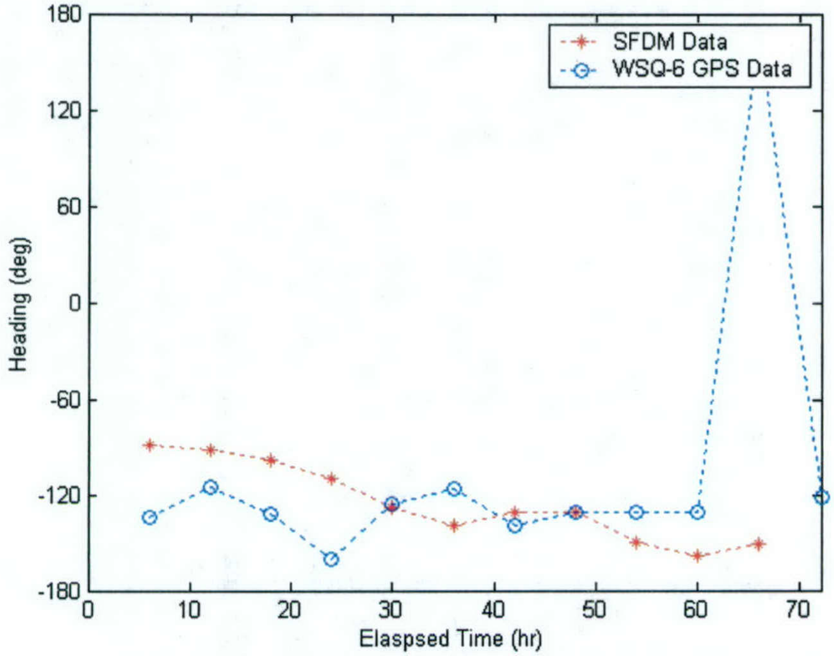


Figure D-3: Buoy 19 – Typical NW Region Heading History

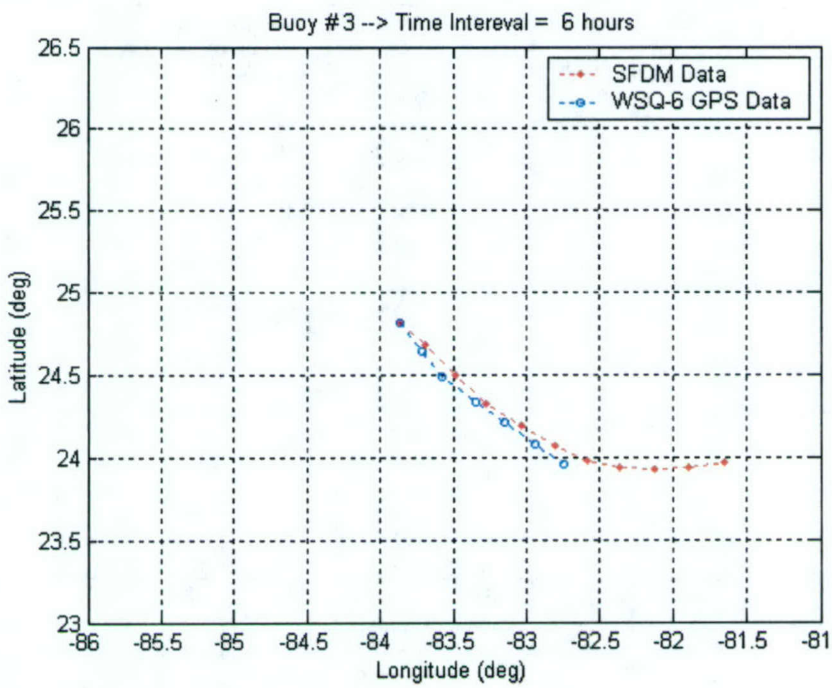


Figure D-4: Buoy 3 – Typical SW Region Trajectory

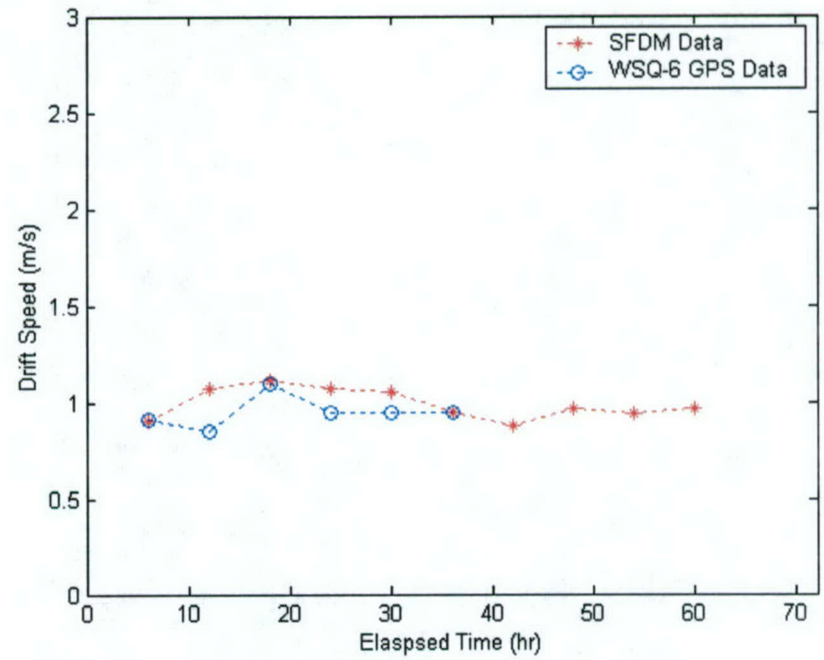


Figure D-5: Buoy 3 – Typical SW Region Velocity History

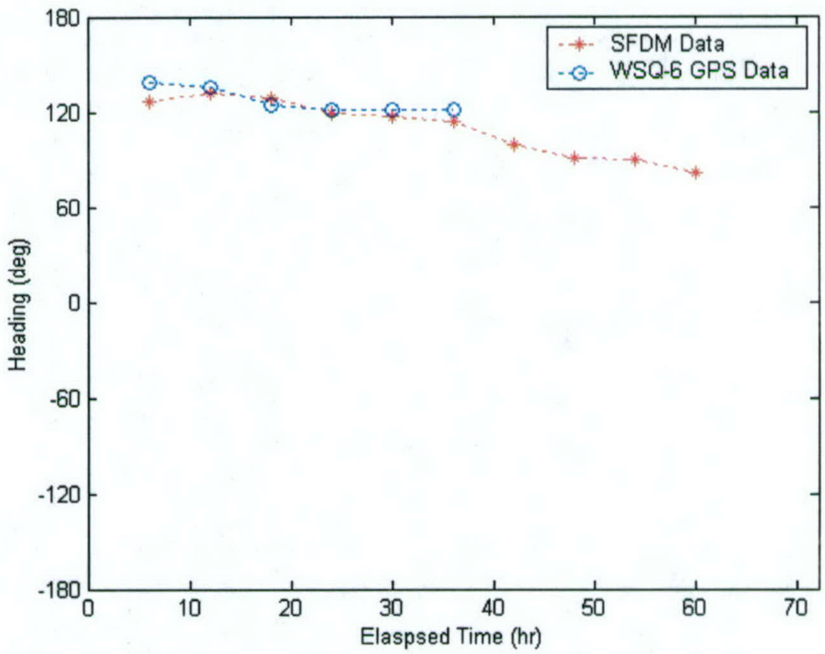


Figure D-6: Buoy 3 – Typical SW Region Heading History

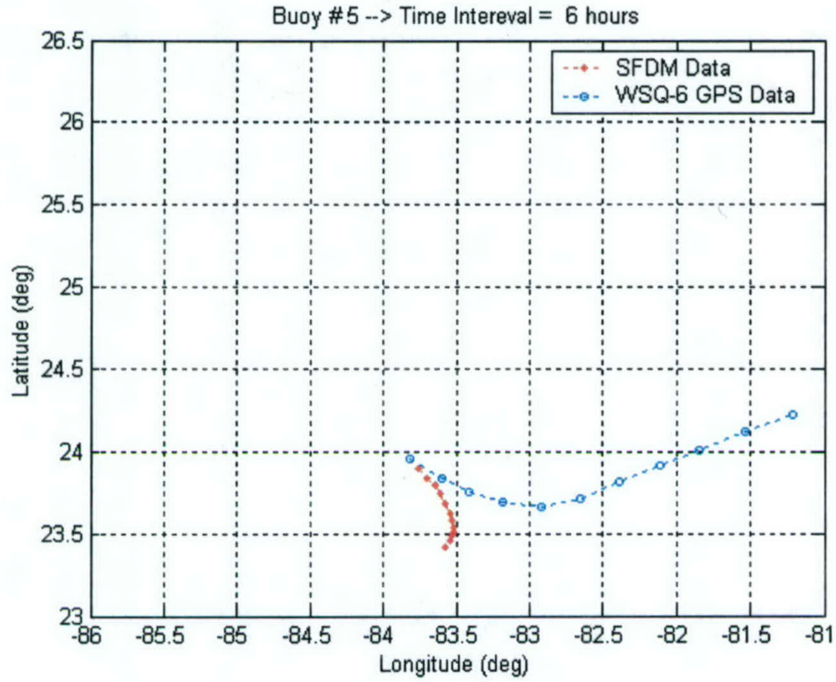


Figure D-7: Buoy 5 Trajectory

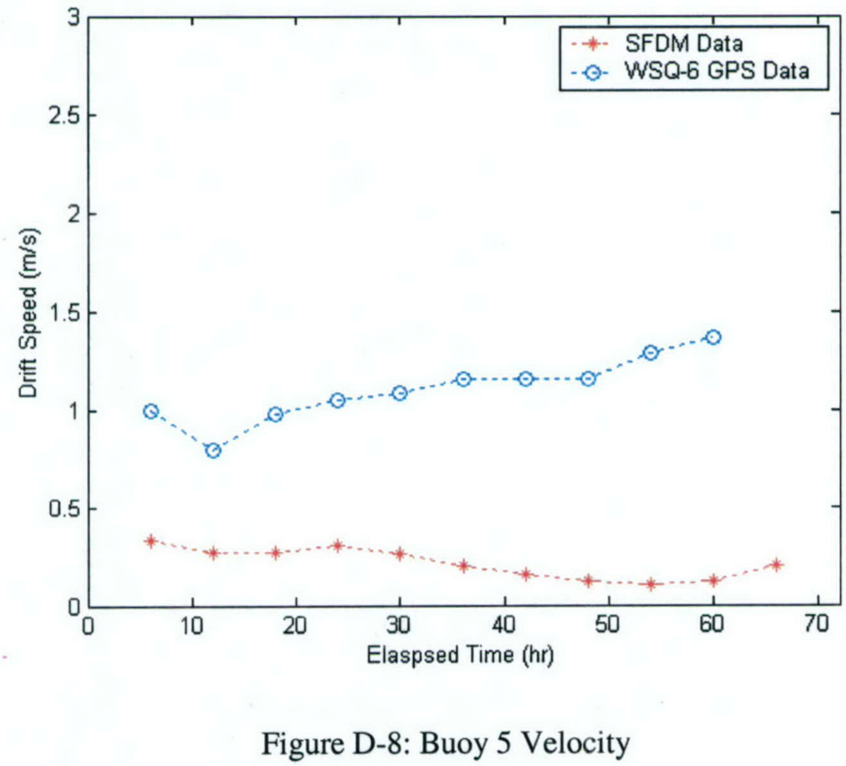


Figure D-8: Buoy 5 Velocity

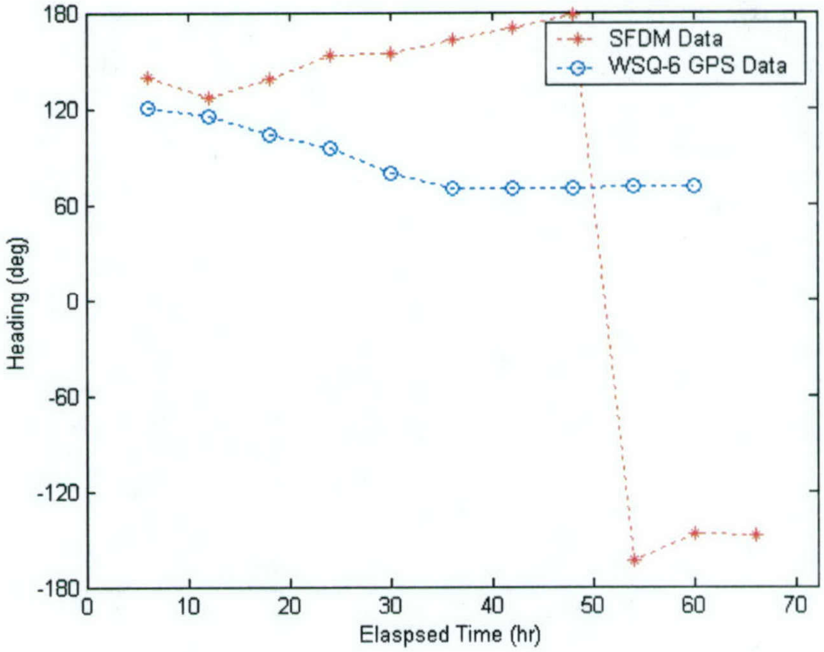


Figure D-9: Buoy 5 Heading

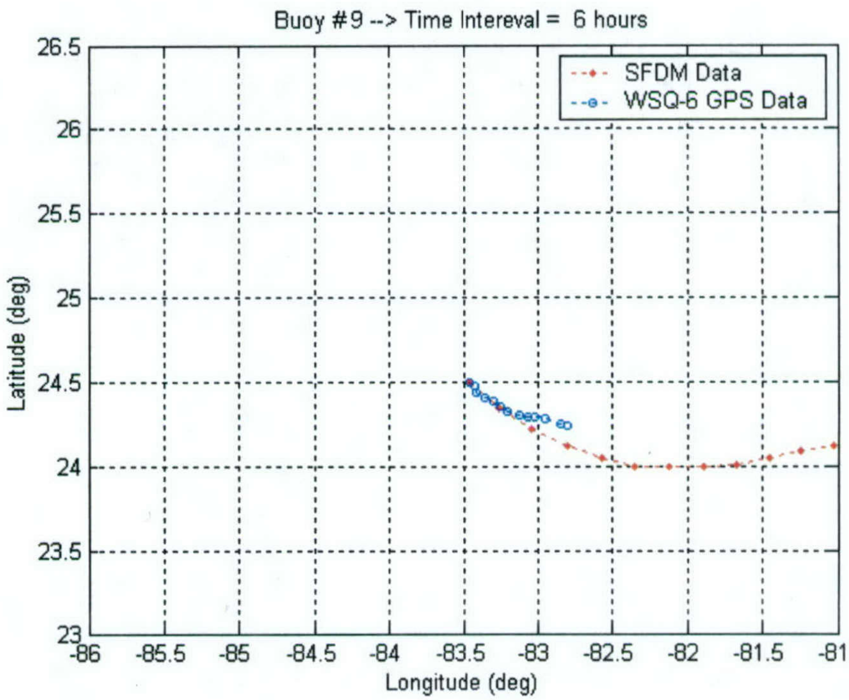


Figure D-10: Buoy 9 Trajectory

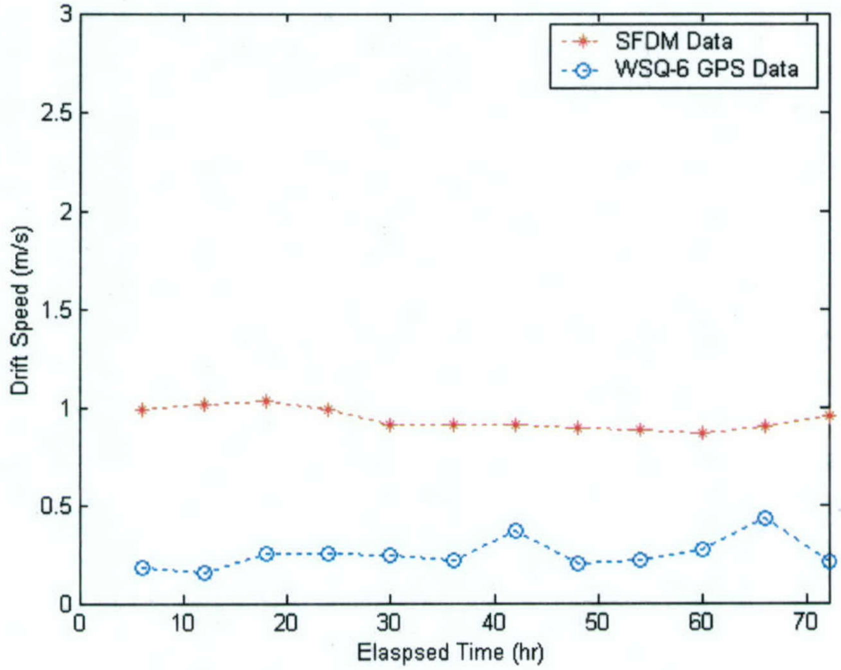


Figure D-11: Buoy 9 Velocity

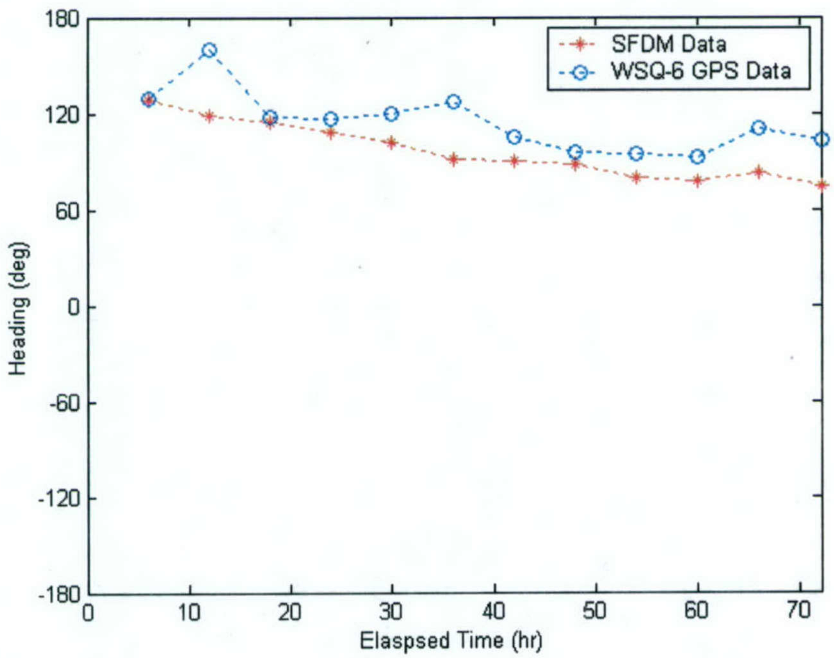


Figure D-12: Buoy 9 Heading

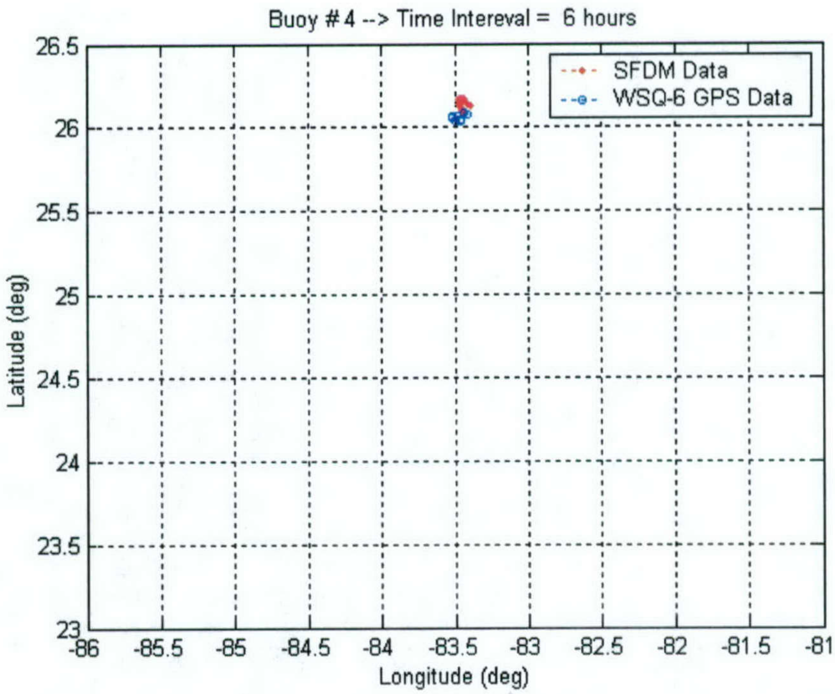


Figure D-13: Buoy 4 – Typical NE Region Trajectory

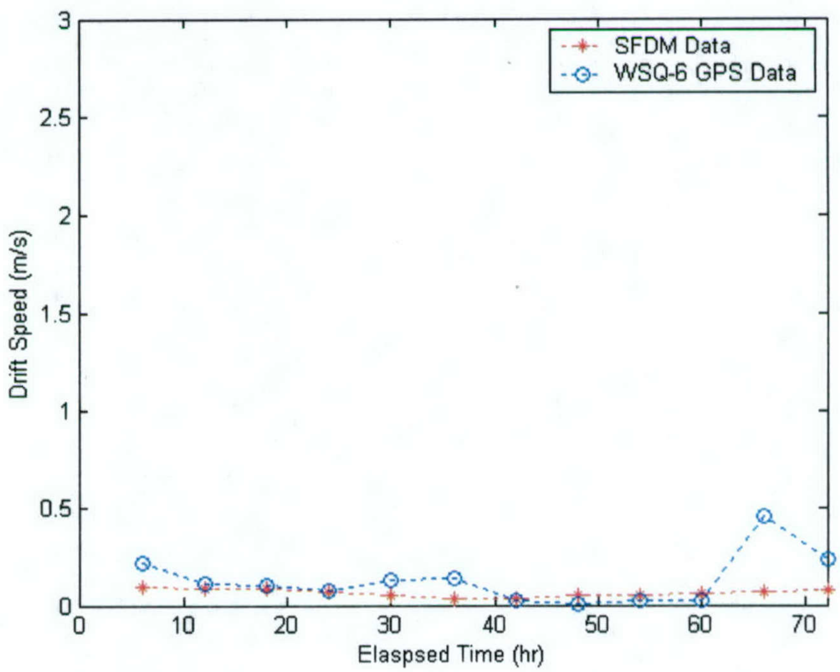


Figure D-14: Buoy 4 – Typical NE Region Velocity

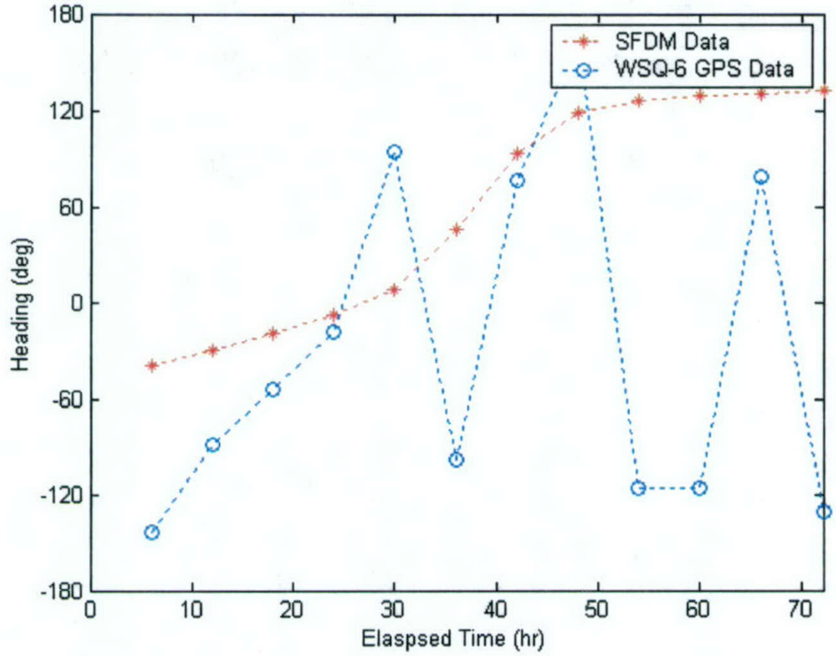


Figure D-15: Buoy 4 – Typical NE Region Heading

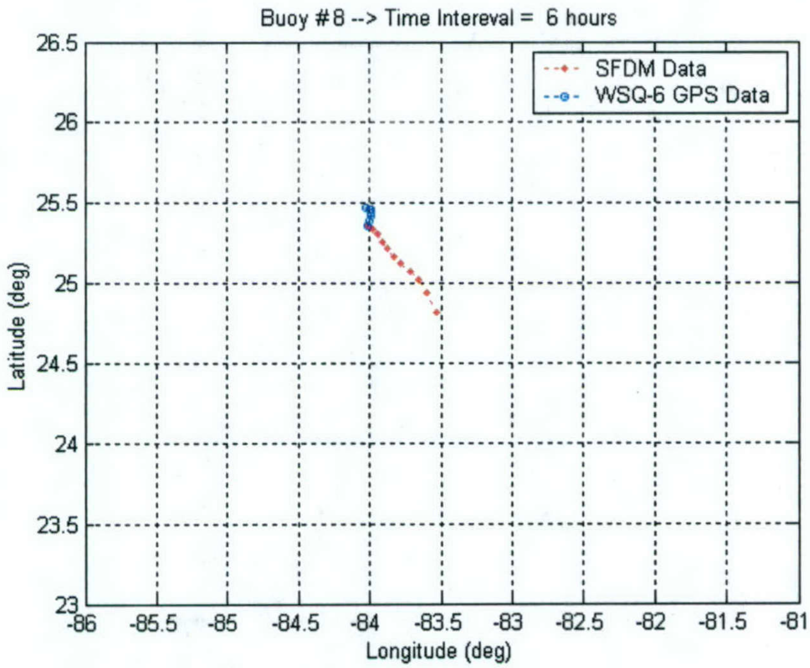


Figure D-16: Buoy 8 Trajectory

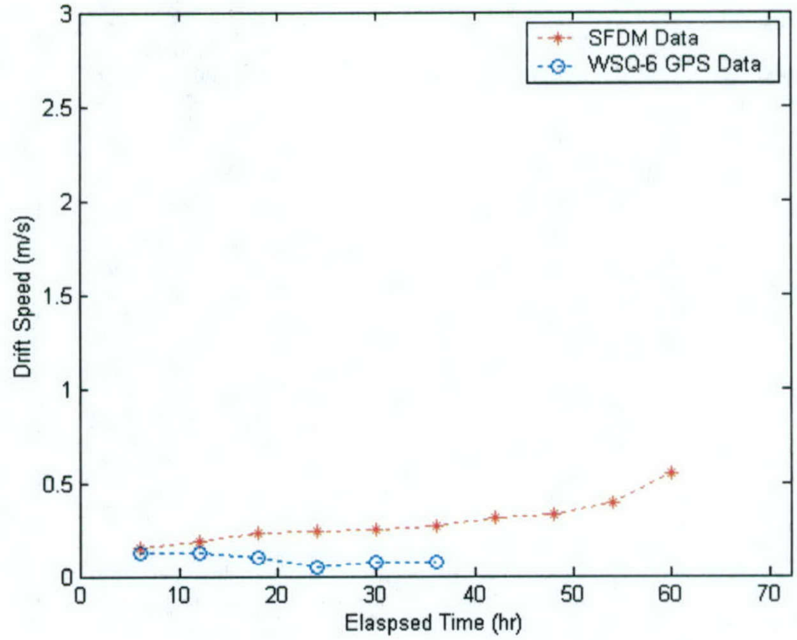


Figure D-17: Buoy 8 Velocity

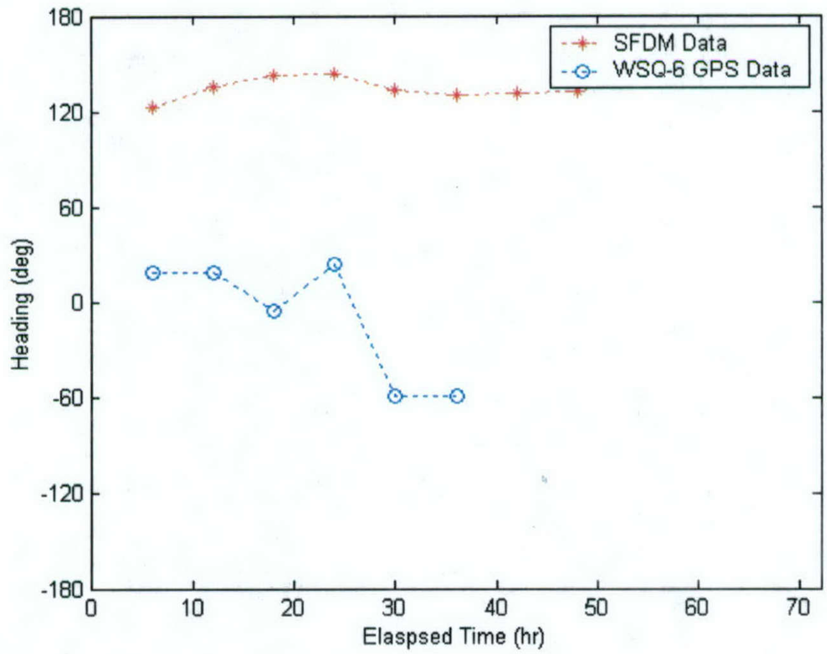


Figure D-18: Buoy 8 Heading

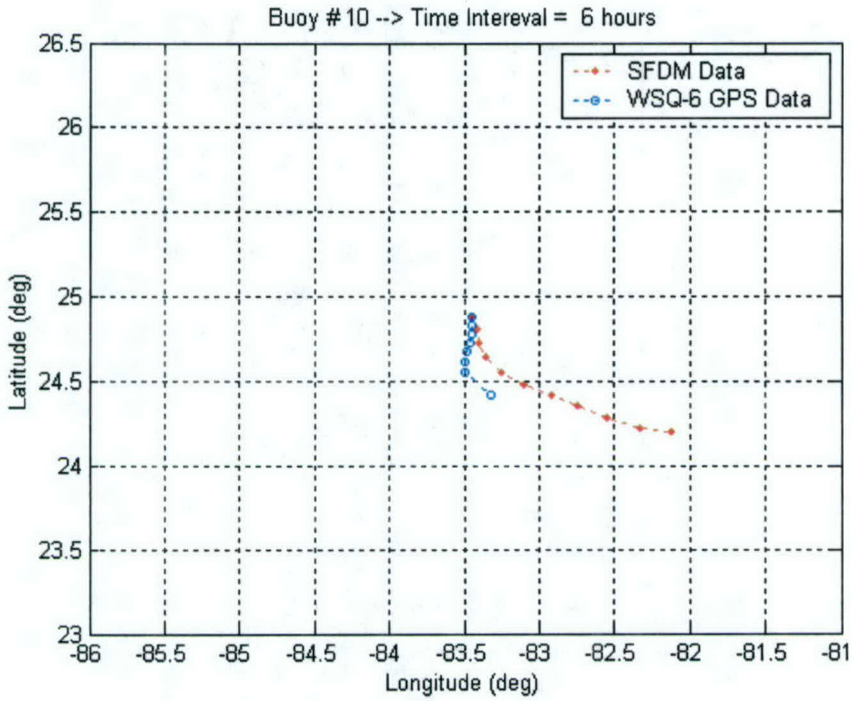


Figure D-19: Buoy 10 Trajectory

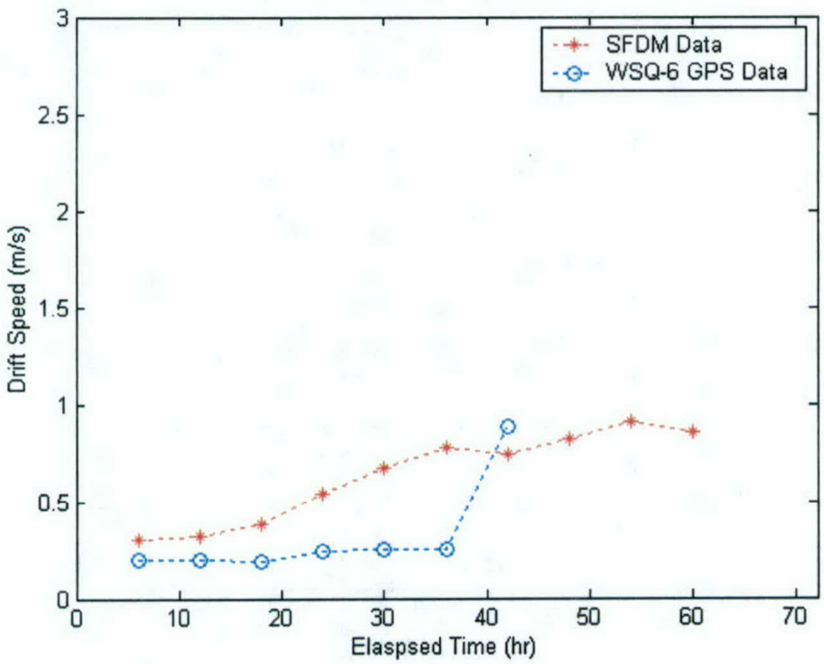


Figure D-20: Buoy 10 Velocity

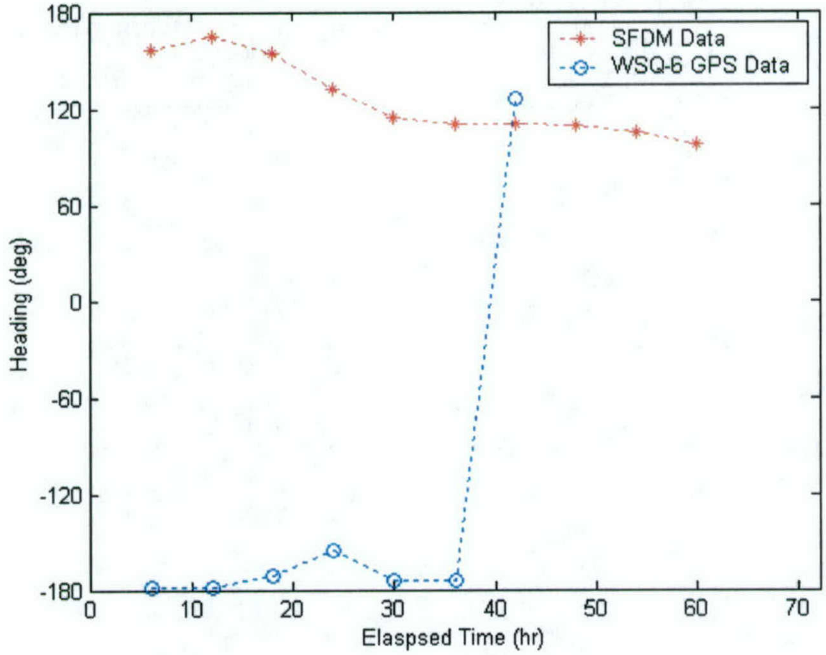


Figure D-21: Buoy 10 Heading

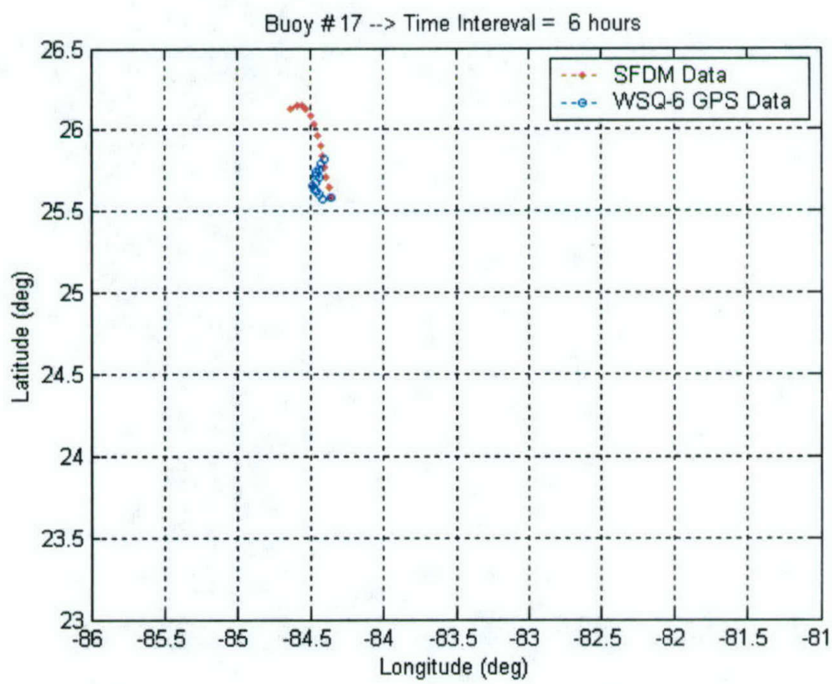


Figure D-22: Buoy 17 Trajectory

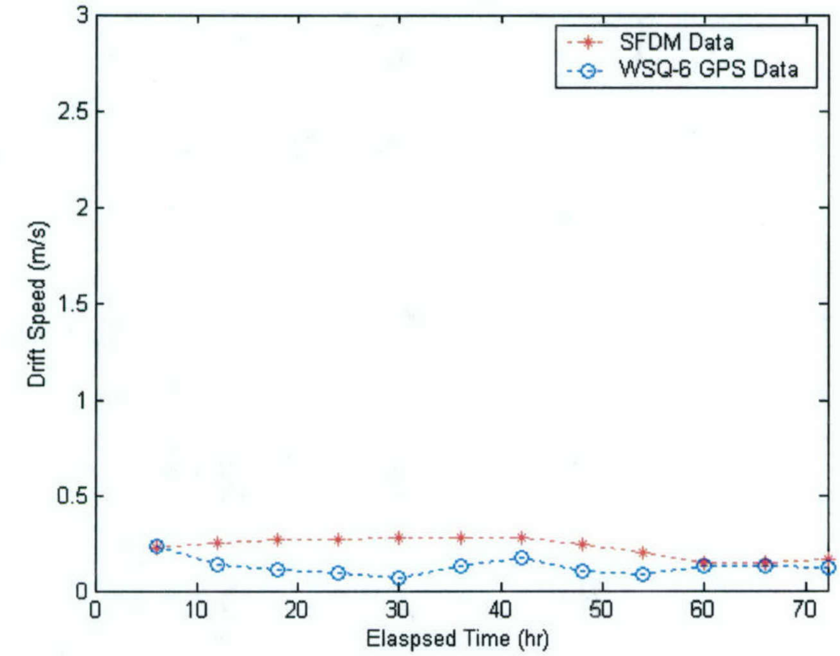


Figure D-23: Buoy 17 Velocity

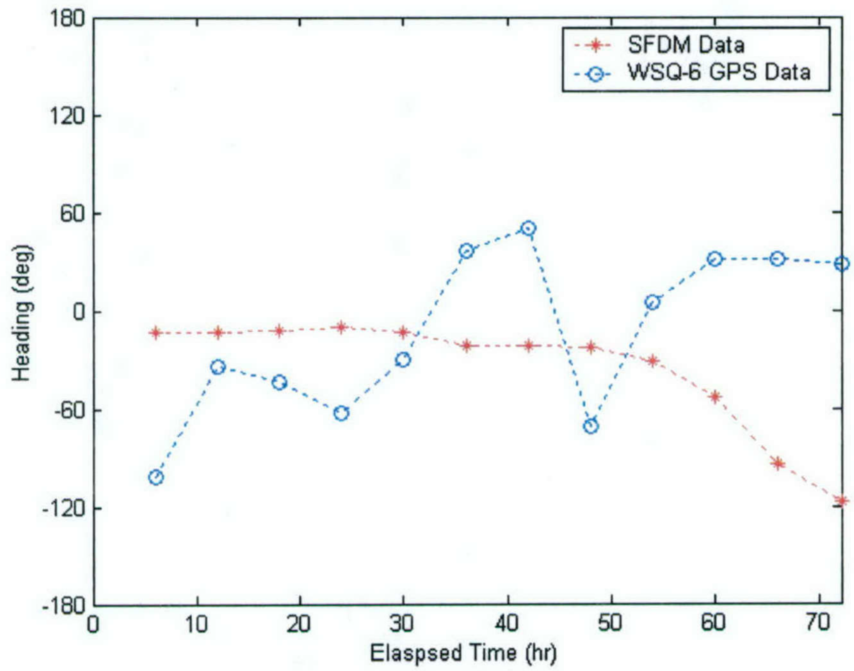


Figure D-24: Buoy 17 Heading

THIS PAGE INTENTIONALLY LEFT BLANK

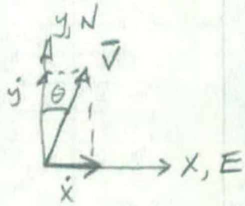
APPENDIX E
DRAG ERROR CALCULATIONS

Total Drag determined using orthogonal N/E components $(\dot{x}, \dot{y})$

$$D_x = \frac{1}{2}\rho C_d A_x (\dot{x})^2 \qquad D_y = \frac{1}{2}\rho C_d A_y (\dot{y})^2$$

$$C_dA_x = C_dA_y = C_dA$$

$$D_{TOT,xy} = \sqrt{D_x^2 + D_y^2} = \frac{1}{2}\rho C_dA\sqrt{(\dot{x})^4 + (\dot{y})^4}$$



Total Drag determined using current vector (V)

$$\bar{V} = V_x\hat{i} + V_y\hat{j} = |V|\sin\theta\,\hat{i} + |V|\cos\theta\,\hat{j} = \dot{x}\hat{i} + \dot{y}\hat{j}$$

$$D_{TOT,V} = \frac{1}{2}\rho C_dA|\bar{V}|^2$$

$$|V|^2 = (\dot{x})^2 + (\dot{y})^2 \Rightarrow D_{TOT,V} = \frac{1}{2}\rho C_dA\left[(\dot{x})^2 + (\dot{y})^2\right]$$

$$\frac{D_{TOT,xy}}{D_{TOT,V}} = \frac{\sqrt{(\dot{x})^4 + (\dot{y})^4}}{(\dot{x})^2 + (\dot{y})^2} = E_D \equiv \text{Drag Error}$$

OR $\dot{x} = |V|\sin\theta$, $\dot{y} = |V|\cos\theta$

$$E_D = \frac{\sqrt{|V|^4\sin^4\theta + |V|^4\cos^4\theta}}{|V|^2} = \frac{\sqrt{|V|^4(\sin^4\theta + \cos^4\theta)}}{|V|^2}$$

$$E_D = \sqrt{\sin^4\theta + \cos^4\theta}$$

$$\sin^4\theta = \frac{1}{8}(3 - 4\cos(2\theta) + \cos(4\theta))$$

$$\cos^4\theta = \frac{1}{8}(3 + 4\cos(2\theta) + \cos(4\theta))$$

$$\sin^4\theta + \cos^4\theta = \frac{1}{8}(6 + 2\cos(4\theta)) = \frac{1}{4}\left[3 + \cos(4\theta)\right]$$

N/E AXIS DRAG ERROR (SFDMV2) 11-19-04 DSH 5

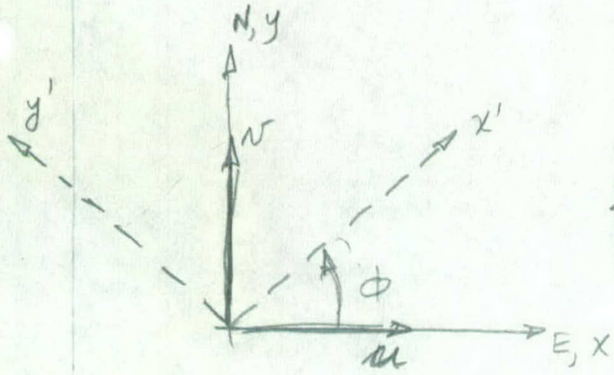
Use weighted mean to rotate axis to "dominant" current direction:

$\phi(z) =$ current direction

$q(z) =$ current energy (intensity) weighting factor

$$\sum_{i=1}^{n} q_i = 1 \qquad \bar{\Phi}_w = \sum_{i=1}^{n} q_i \phi_i \qquad n \equiv \text{\# of current depth strata}$$

$$q_i = \frac{V_i^2}{\sum_{i=1}^{n} V_i^2} \quad \text{where} \quad V_i^2 = u_i^2 + v_i^2$$



Rotate u, v axis (x, y) dominate axis (x', y')

$$\begin{Bmatrix} u' \\ v' \end{Bmatrix}_i = [A] \begin{Bmatrix} u \\ v \end{Bmatrix}_i$$

for each depth

$$A = \begin{bmatrix} \cos\phi & +\sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}$$

so $u_i' = u_i \cos\phi - v_i \sin\phi$

$v_i' = u_i \sin\phi + v_i \cos\phi$

DISTRIBUTION:

NAVAIRWARCENACDIV (4.5.14.2/Hammond), Bldg. 2185, Room 1160-C3 22349 Cedar Point Road, Patuxent River, MD 20670-1161	(4)
Office of Naval Research (3.2 ASW, Dr. Dave Johnson; 322OM, Manuel Fiadiero; 321MS Michael Traweek) 800 North Quincy Street, Arlington, VA 22217-5660	(3)
NAVAIRSYSCOM (AIR-5.1V), Bldg. 304, Room 120 22541 Millstone Road, Patuxent River, MD 20670-1606	(1)
NAVAIRSYSCOM (AIR-5.1), Bldg. 304, Room 100 22541 Millstone Road, Patuxent River, MD 20670-1606	(1)
NAVAIRWARCENACDIV (7.2.5.1), Bldg. 405, Room 108 22133 Arnold Circle, Patuxent River, MD 20670-1551	(1)
NAVTESTWINGLANT (55TW01A), Bldg. 304, Room 200 22541 Millstone Road, Patuxent River, MD 20670-1606	(1)
DTIC 8725 John J. Kingman Road, Suite 0944, Ft. Belvoir, VA 22060-6218	(1)

UNCLASSIFIED

UNCLASSIFIED